パソコンプログラマのためのクリエイティブマガジン

1991年2月1日発行(毎月1回発行) 第1巻 第2号 通巻2号

月刊プログラマーズ・ページ

Programmer's Page

MONTHLY MAGAZINE FOR PROGRAMMERS

1991 2 FEBRUARY

定価980円

創刊記念プレゼント
HANDY98他

SE
Shoeisha Co., Ltd.

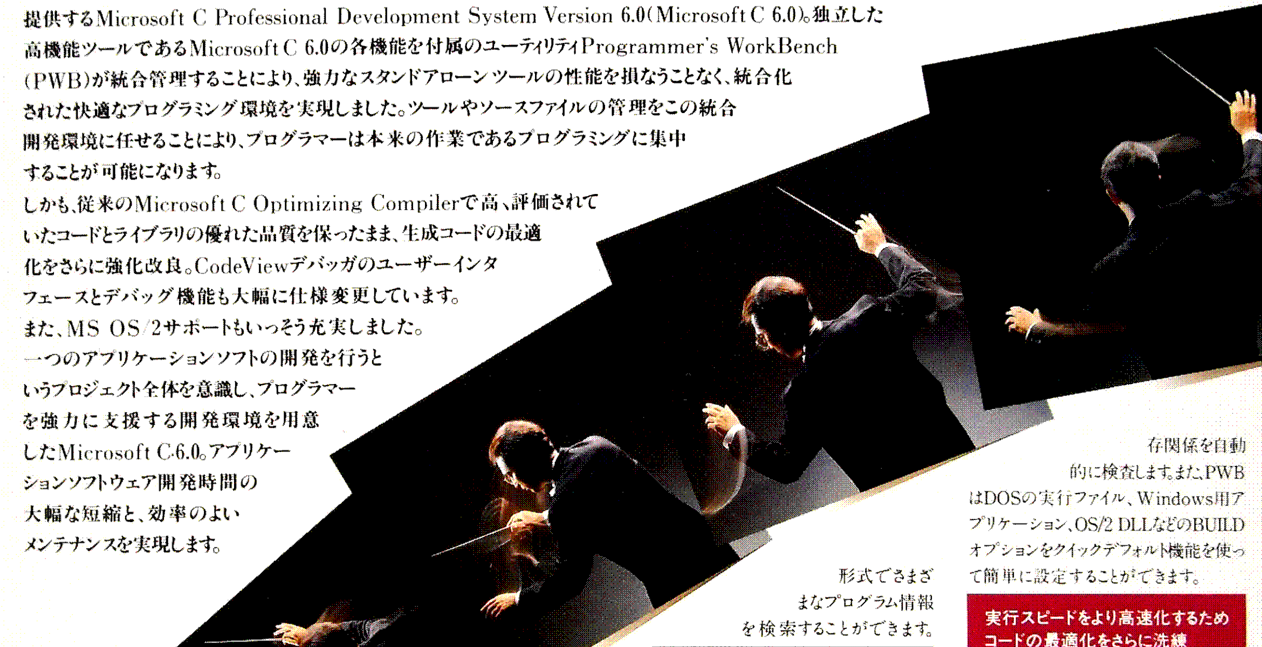
これが、Cの正統。

アプリケーションソフトウェアの開発に携わるプロフェッショナルユーザーの方々のために、強力で柔軟な統合開発環境を提供するMicrosoft C Professional Development System Version 6.0(Microsoft C 6.0)。独立した高機能ツールであるMicrosoft C 6.0の各機能を付属のユーティリティProgrammer's WorkBench(PWB)が統合管理することにより、強力なスタンドアローンツールの性能を損なうことなく、統合化された快適なプログラミング環境を実現しました。ツールやソースファイルの管理をこの統合開発環境に任せることにより、プログラマーは本来の作業であるプログラミングに集中することが可能になります。

しかも、従来のMicrosoft C Optimizing Compilerで高い評価されていたコードとライブラリの優れた品質を保ったまま、生成コードの最適化をさらに強化改良。CodeViewデバッガのユーザーインタフェースとデバッグ機能も大幅に仕様変更しています。

また、MS OS/2サポートもいっそう充実しました。

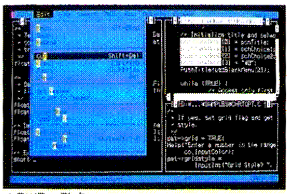
一つのアプリケーションソフトの開発を行うというプロジェクト全体を意識し、プログラマーを強力に支援する開発環境を用意したMicrosoft C 6.0。アプリケーションソフトウェア開発時間の大幅な短縮と、効率のよいメンテナンスを実現します。



PWBを中心に生産性を強力に向上

●強力なエディタ機能

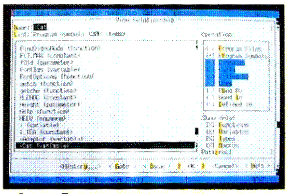
QuickC 2.0と同様のユーザーインタフェースを持つProgrammer's WorkBench(以下PWB)のエディタ機能。しかも、キーの割付変更や編集機能の拡張などの豊富なカスタマイズ機能、独自のマクロ言語を使った編集機能の定義、レコードマクロ、同時に8つのウィンドウをオープンできるマルチウィンドウ機能を装備しています。



▲PWBエディタ

●Source Browser機能

PWBは作成中のアプリケーションソフトウェアの構造を解析するSource Browser機能を備えています。これにより、対話形式でさまざまなプログラム情報を検索することができます。



▲Source Browser

●Microsoft C Advisor

PWBはQuickC 2.0同様に強力なオンラインリファレンスシステムを装備しています。

▲Microsoft C Advisor

●プログラムメンテナンス機能

PWBは、プログラムリストを使って簡単にMAKEファイルを自動作成。プログラムリストは、MAKEファイルを作成する際に、インクルードファイルを含むファイル依存関係を自動的に検査します。またPWBはDOSの実行ファイル、Windows用アプリケーション、OS/2 DLLなどのBUILDオプションをクイックデフォルト機能を使って簡単に設定することができます。

実行スピードをより高速化するためコードの最適化をさらに洗練

●グローバルな最適化の実現

Microsoft C 5.1で行われていた"ループレベル"での最適化から、関数全体に対してグローバルな最適化を実現するように進化。また、変数を自動的にレジスタ変数に割り当てるようにしています。

●ローカルコードの最適化の実現

数式を手作業で最適化するときの"経験則"を導入。shift式やswitchステートメントを有効に最適化できます。

●最適化レベルの管理の強化

コンパイル時の指定以外にプラグマによっても最適化の管理ができ、特定の関数を思い通りのレベルで最適化することができます。

コードのパフォーマンス向上と開発時間の短縮を同時に実現する新しい言語仕様

●basedポインタ

nearポインタのコンパクトさとfarポインタの機能を合わせ持つポインタです。nearポインタと同様に16ビットのアドレスしか

●以下のシリーズ対応につきましては、直接各社様宛お問い合わせください。

FM Rシリーズ対応（富士通株式会社より発売予定）お問い合わせ先:03-3730-3140

Panacom Mシリーズ対応（松下電器産業株式会社より発売予定）お問い合わせ先:0120-158756

ありがとうキャンペーン 実施中

期間：'90年12月1日〜'91年1月31日

この期間にマイクロソフト製品をご購入になり、登録された方の中から抽選で3,000名様に、マイクロソフトグッズをプレゼント。

詳しくは、キャンペーンポスターのあるお店で。

LIMは、Lotus/Intel/Microsoftが提唱するEMSの方式です。またEMSはExpanded Memory Specificationの略語です。
ATOKは㈱ジャストシステムの、Lotusは米国ロータスデベロップメント社の、Intelは米国インテル社の登録商標です。
Microsoft、MS、MS-DOS、CodeViewは、米国マイクロソフト社の登録商標です。MS-KANJI APIは、マイクロソフト株式会社が推奨する日本語入力システムの仕様です。

必要とせず、farポインタのように64Kを超えるメモリセグメントの参照を可能にしました。

●プロシージャコールの最適化

新しいプロシージャコール_fastcallは、関数間の引数の受渡しの際に直接レジスタを利用することで実行速度を大幅に向上させます。

●インラインアセンブリ機能

Microsoft C 6.0のコンパイラは、インラインアセンブラを統合しています。もちろん、アセンブルの際には、Microsoft Macro Assemblerは必要ありません。

●TINYメモリモデルの追加

マイクロソフト社の高級言語ではじめてTINYメモリモデルをサポート。このメモリモデルを利用することにより、MS-DOS上で動作するCOMファイルをC言語で作成することができます。(ただし、グラフィックスライブラリ、BIOSライブラリは使用できません)

●ライブラリの強化

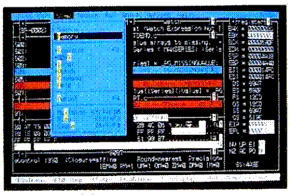
BIOSライブラリと強化されたグラフィックスライブラリを装備。もちろん、FONTライブラリ、チャートライブラリも追加されました。

最新のCodeView 3.0でデバッグ機能を強化しユーザーインタフェースを改善

●容量の増大

CodeView自身のコードをEMSに追い出すことにより、より大きなターゲットプログラムのデバッグが可能になりました。

●データアクセスの拡大



▲CodeView 3.0

ローカルウィンドウをサポート。また、ローカルウィンドウやウォッチウィンドウを利用し、配列やネスト構造体の内容を展開表示したり、ポインタが指しているアドレスの内容を表示したりすることが可能です。さらにデバッグ中の8087/80287/80387コプロセッサのステータスを表示するためのコプロセッサウィンドウや、指定した変数や式の現在の値を即座に表示するクイックウォッチ機能が追加されました。

●ユーザーインタフェースの改良

ユーザーインタフェースを完全に再設計し、QuickシリーズやPWBと同様のユーザーインタフェースを実現。各ウィンドウは独立したまま共通の操作環境を持ち、各ウィンドウ単位での縦横のスクロールや内容の印刷も可能です。

●ダイナミックリプレイ機能

デバッグ途中のプログラムに変更を加えた場合、変更した時点までに入力したすべての情報をプログラムの先頭から自動的に再生するダイナミックリプレイ機能を装備しています。

●PWBとの統合

CodeView 3.0は、従来のCodeViewと同じく独立したツール。しかし、PWBとの組み合せを考えた設計を行い、PWB側であらかじめ設定しておくことが可能です。また、CodeView3.0のデバッグセッションの情報も自動的に保存されますから、いったんPWB側に戻った後でもウィンドウの設定、ウォッチ変数、ブレークポイントなどの情報は自動的に再設定されます。

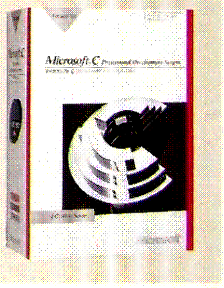
■必要システム

[NEC PC-9800シリーズ PC-98シリーズ]

- ●ハードウェア:80286、80386をCPUに持つNEC PC-9800シリーズ、またはPC-98シリーズ(ただし、ハイレゾリューションモードを除く)
- ●OS:MS-DOS 3.1以上、もしくはOS/2 1.1A以上
- ●必要メモリ:640Kバイト(MS-DOS)、空きメモリ1Mバイト以上(OS/2)
- ●ディスク装置:1Mバイト容量の1台以上のフロッピーディスク装置と1台以上のハードディスク装置(ハードディスクの空き容量は最低約6Mバイト以上、最大20Mバイト必要)
- ●推奨システム:1Mバイト以上のLIM4.0対応のEMSメモリ(MS-DOS)、1Mバイト以上のキャッシュディスク(MS-DOS)、マウス
- ●使用可能な日本語入力システム:MS-KANJI API仕様の日本語入力システム(ATOK6、ATOK7の場合、付属のドライバで対応可能)

■実行ファイルのターゲットシステム

- ●ハードウェア:NEC PC-9800シリーズ、PC-98シリーズ(ただし、グラフィックスライブラリを使用する場合は、PC-98LTおよびハイレゾリューションモードを除く。またBIOSライブラリを使用する場合は、PC-98LT/XAを除く)
- ●OS:MS-DOS 2.11以上、もしくはOS/2 1.1以上
  Microsoft Windows 3.0以上
  (実行ファイル作成には日本語MS-WINDOWSツールキットが必要。MS-DOSとMS OS/2のBOUND実行ファイルを作成するときは、日本語MS OS/2ソフトウェア開発支援ツールが必要)
- ●必要メモリ:システムと作成プログラムに依存



Microsoft® C

Professional Development System Version 6.0

NEC PC-9800シリーズ対応 標準価格 98,000円(消費税別) 好評発売中

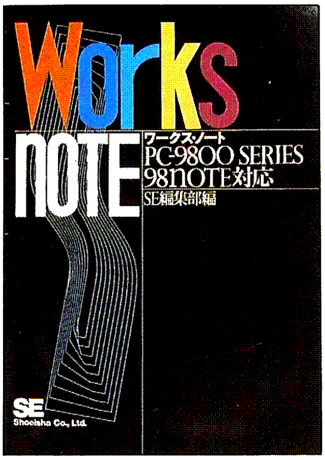
マイクロソフト株式会社 〒160 東京都新宿区西新宿7-5-25 Kビルディング 電話(03)3363-1201

<資料請求番号P02>

SEパソコンユーザ

Microsoft® Worksガイドブックの決定版！

ワークス・ノート



◆SE編集部編
◆定価1,500円

Worksは、ワープロ、表計算、データベース、通信の四つの機能を統合した、行動派のためのやさしい知的生産ツールです。 洗練されたコマンド体系は、パソコンを楽しいアイデア作りの道具に変身させます。本書はWorksのコマンド体系・機能の解説はもとより、実用的でアイデアに満ちた事例を満載、4つの機能の連携による1×4＞4の美味しさを紹介します。

トレーニングブックシリーズ

アプリケーションソフトの
超ていねいな解説書を
お求めの方は、是非一度お試し下さい。

マルチプランVer.4,1 トレーニング

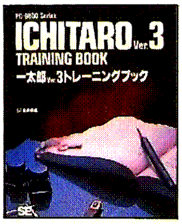
●山口博幸 著　●定価2,000円

ステップアップ方式を取り入れて、実践的かつ豊富な画面図によりわかりやすく解説。

一太郎Ver.4 トレーニングブック

●SE編集部編　●定価1,600円

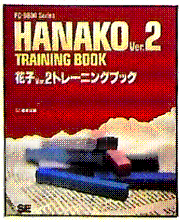
強化された一太郎の機能をあますところなく紹介。



一太郎Ver.3 トレーニングブック

●SE編集部編　●定価1,600円

根強い人気を維持するVer.3の機能を徹底的に解説。



花子Ver.2 トレーニングブック

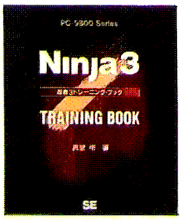
●SE編集部編　●定価1,600円

新機能を加えた花子を多彩な図版によって解説。

花子 トレーニングブック

●SE編集部編　定価2,060円

花子の機能と操作を、丁寧に解説したベストセラー。



Ninja 3 トレーニングブック

●真堂彬 著　●定価1,800円

人気者のカード型データベースの利用法をわかりやすく解説。

日本語EXCELトレーニングブック

●SE編集部編　●定価2,060円

Mac用のEXCEL使用法を多彩な例をあげて解説。

シルエットトレーニングブック

●SE編集部編　●定価1,648円

概念から描き方の例まで豊富な図版で解説。

—のためのこの一冊。

パソコン文化のトレンドを探るバラエティブック

PDS白書

シリーズ13弾

公開ソフトウェアをめぐる考察とレビュー

◆SE編集部編
◆定価1,500円

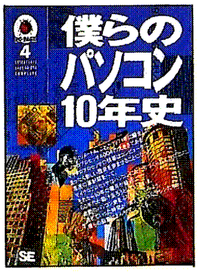
自分で作ったプログラムを公開してソフトウェアの資産の共有化を図ろうとする「PDS（パブリックドメインソフトウェア）」の運動は、パソコンが生み出した文化の最も豊饒（ほうじょう）な部分である。ともすれば「便利でタダ」という形で語られがちな「PDS」について、その歴史的な背景、現状の分析、今後の方向性などの側面から考察する。また現在流通している公開ソフトウェア40本を駆使して構築したモデルシステムもあわせて紹介。

毎月7日は PC-PAGEの日

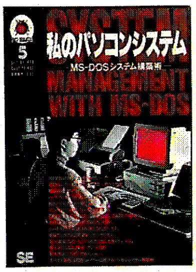
標準秋葉原ガイドブック
パソコンユーザーのための秋葉原電気街ガイド。
定価1,200円

一太郎全ファイル解説
一太郎Ver.4のすべてのファイルについて解説。
定価1,200円

林檎百科
〜マッキントッシュクロニクル〜アップル社の歴史と話題をまとめた読み物。
定価1,200円



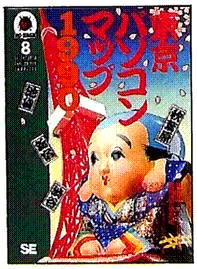
僕らのパソコン10年史
パソコン発展の年代史＋9人の著者の語り。
定価1,200円



私のパソコンシステム
〜MS-DOSシステム構築術HDDの使い方、システム構成の実例集。
定価1,200円

パソコンブックガイド1990
業界著名人が選んだ目的別のブックス100冊。
定価1,200円

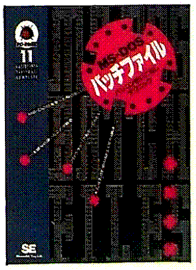
BBSの歩き方
主要ネットワークのマップ＋コマンドリファレンス。
定価1,200円



東京パソコンマップ
東京のパソコンスポットを網羅した便利帳。
定価1,200円

MS-DOS3.3全ファイル解説
MS-DOS3.3の全ファイルの意味と利用法を解説。
定価1,200円

パソコンを思想する
選ばれた15人の「論者」たちが社会的・哲学的側面からパソコンを考察します。
定価1,200円



MS-DOSバッチファイル
パソコンをもっと"使いこなす"ための必読の書。
定価1,500円

プロフェッショナルのパソコン利用術
パソコンをクリエイティブに活用しているプロフェッショナルのシステムとそのノウハウを具体的に紹介する
定価1,200円

PC-PAGE14

さて気になる次号のタイトルは……パソコンとMS-DOS

MS-DOSは、偶然の産物ではない、必然は必然を生んで、今も進化し続けている。本書は、MS-DOS誕生から今日までの歴史的経緯をたどりながら、パソコン、とりわけCPUとの関係の中から、MS-DOSの役割とその基本機能の仕組みを解説する。

SE
株式会社 翔泳社
東京都千代田区平河町2-4-14 平河町KSビル 〒102
PHONE 03-3263-0447 FAX 03-3263-4526

〈資料請求番号001〉

Brief 日本語

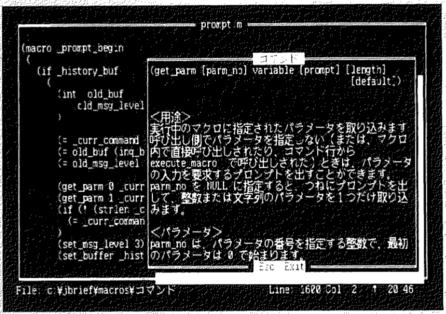
■Full Screen Editor

IBM PCでトップセールスに輝く
プロフェッショナル エディター「BRIEF」

ブリーフの高機能 その①

■「メモ」「書式」「スケジュール」etc……

編集中も表示可能、マクロの応用機能。



メモ画面

対応機種一覧

マクロは「工夫」を実現します!

たとえば、マクロでオーバーラップ・ウインドウを表示することができます。表示だけでなくウインドウの中に書き込むこともできます。もちろんアンドウも使用できます。これを使えばいろいろなことができそうです。
編集中に電話番号やちょっとしたメモが必要になったりします。そのたびに住所録やメモ帳を取り出すのは面倒なものです。そんなときはオーバーラップ・ウインドに表示させましょう。メモを見たり、書き込んだり、机の上を探し回る無駄な時間は最小限にしたいものです。

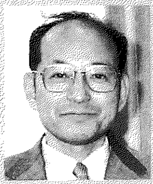
使わない方にも便利なマクロです!

「自分で作らないからマクロ機能はいらない!」と決めつけないでください。公開されたマクロを手にいれれば新しい機能を使うことができるのです。作らなくても機能を増やせることになります。このため弊社では、公開マクロのユーザー相互利用を推進しております。
Wordstar用のキー・アサインや行のソート処理など魅力的なマクロを公開しています。
Briefマクロプログラミング・コンテスト(8月末締切)の応募マクロも公開する予定です。公開マクロについてはBriefユーザー・サポート係にお問い合わせください。

USER'S VOICE

Briefとは長い付き合いになると思う。

私どもの実験は非常に厳しい条件で行うためデータの記録には迅速かつ正確さが要求されます。記録に漏れがあればデータの分析ができなくなり、全ての努力が無駄になります。このためマクロを利用し、記録項目をガイドに従って入力させることにしました。また、入力ミスにはアンドウで対処することで、迅速かつ正確な入力が実現できました。
日本語Briefとこれから先は長い付き合いとなると思います。



東京理科大学 工学部経営工学科
富澤儀一助教授

●革命的マルチマシン・エディター

東芝・NEC・AX協議会・IBM・富士通の5機種のパソコンで動作します。

●300回ものアンドゥ

各ファイルごとに300までの操作を元に戻すことができますので、やり直しの面倒を気にせず、安心して編集できます。

●55ウィンドウ、無限のファイル数

ウィンドウは縦横自由に55まで開くことができます。また同時に編集できるファイルの数や大きさには制限がありません。

●プログラミング

◆自動コンパイル、エラージャンプ◆スワップアウト◆自動インデント、自動テンプレート

●安全編集

Briefは、一定時間キーを操作しない場合、その時点の編集内容をファイルに自動保存しますので、編集中に電源が切れても安心です。

オピニオンリーダー誌のこの評価

- プログラミング向きのエディターを探している人は、もう探す必要はない。Jerry Pournelle(BYTE)
- Briefはプログラマーに最適なだけでなく創造力をかきたてる何物かがある。Computer World
- ほとんど完ぺきにカスタマイズできるだけの拡張性がある。Computer language
- まるで夢のようなアンドゥだ おもわず間違うのを期待してしまいそうだ。The C Users Journal

■主な機能

◆バッファ書出 切換 削除 リスト表示◆ファイル名の指定 表示 変更◆バックアップ保存◆特殊文字入力◆2行を1行に連結◆ファイル単位の挿入◆増分検索◆各種マーク 始・終点の入換◆ブロックのカット コピー ペースト 印刷 保存 左右へのシフト◆対応する括弧へ移動◆C言語のコメント化 非コメント化◆DOSコマンドの実行 ◆キー操作記憶/実行 ◆入力ヒストリー◆英文テキスト整形◆タブのスペース挿入◆拡張子毎に異なった設定ができるタブ◆検索置換パターンの画面からの入力◆英字の大小文字変換 その他

定価 48,000円(消費税は含まず)

■全機種対応パッケージ

5"2DD 3.5"2DD一括提供(MS-DOS3.1以降対応)
上記以外のメディアは注文販売となります。
英語用DOSでは使用出来ません。

日本システム株式会社

●お問い合せは 東京都立川市曙町1-18-2一清ビル立川事業所Briefユーザー・サポート係
TEL.0425-23-4469 FAX.0425-27-6127

Programmer's Page
MONTHLY MAGAZINE FOR PROGRAMMERS

パソコンプログラマのための
クリエイティブマガジン
月刊プログラマーズ・ページ

1991 FEBRUARY No.2

CONTENTS

特集
Windowsの時代である!

窓の歴史 20

Windowsプログラミングを徹底解説・大解剖
Windows 3.0がパソコン環境に新しい展開をもたらす 26
Windows プログラミング講座1
Windowsシステムとプログラムの条件 36
Windows プログラミング講座2
Windows 3.0＋MS-C 6.0による開発環境構築法 永井知己 44
Microsoft Windows Device Development Kit (DDK) 西川和久 52
Windows プログラミング講座3
Windows基本プログラム大解剖 永井知己 54
Windows プログラミング講座4
実践的Windowsプログラミング技法 滝口 力 69
Windows プログラミング講座5
新しいWindowsプログラミング
WindowsMAKER ver. 3.01 西川和久/Asymetrix ToolBook/Actor 3.0 79

Illustration : NAOKI SATO

CONTENTS

INFORMATION PAGE1 1
NEWS & NEW PRODUCTS 2
佐々木哲のN. Y. システム通信 12
西川和久の希望的観測 14
NEXT STAGE 16

パソコン博物館 第2回
システムとして売り出されたパソコン/APPLE II 井戸禎光 17

プログラマの道具学 第2回
電卓 祐安重夫 90
オダジマタカシのただいま悪戦苦闘中
小田嶋隆 92

Macintoshで考えたこと
Personalから Intimateへ 大座畑重光 94
実践C塾 第2回
メモリモデルとオーバーレイ 水居 浄 98
98活用アセンブラ技法/ビデオシステム編 第2回
半角外字フォントの読み込みと設定 星野操 112
プログラマのためのハードウェア再入門2
ディスプレイ 鎌田正寿 122
プログラム「遊」工房 弐乃巻
モアレ模様 谷口典生 127

今月の受験情報
情報処理技術者試験の動向 赤松高文 133

INFORMATION PAGE2 137
BookReview「コンパイラI. II」中村正三郎 138
本のアンテナ 古田島義和
Programmer's Program RCS 140
プログラマ登場②片桐明 141
オンラインソフト採集帳2「FD」 142
HEALTHY? UNHEALTHY?
コンピュータの法律 熊谷信太郎 144
アキバ短信
バーチャルスペース・ガイド 146
創刊記念プレゼント 148
ドラゴン・ポスト
クイズ・ランダムスクリーン 150

付録
MS-Windows 3.0関数一覧

表紙イラスト：日暮修一
表紙AD：佐々木幹夫(SE)

広告索引　　資料請求番号

マイクロソフト（株）……………………………P02
日本システム（株）………………………………002
丸文（株）…………………………………………P03
三菱電機（株）……………………………………P04

人材募集広告

ニューソンコンピュータ（株）
タム（株）
（掲載順）

翔泳社　　書籍案内

●PC-PAGEシリーズ

PC-PAGE 1　パソコン版標準秋葉原ガイドブック
A5判　定価1200円（本体1165円・税35円）
PC-PAGE 2　一太郎Ver4全ファイル解説
A5判　定価1200円（本体1165円・税35円）
PC-PAGE 3　林檎百科-マッキントッシュクロニクル-
A5判　定価1200円（本体1165円・税35円）
PC-PAGE 4　僕らのパソコン10年史
A5判　定価1200円（本体1165円・税35円）
PC-PAGE 5　私のパソコンシステム
A5判　定価1200円（本体1165円・税35円）
PC-PAGE 6　パソコンブックガイド1990
A5判　定価1200円（本体1165円・税35円）
PC-PAGE 7　BBSの歩き方
A5判　定価1200円（本体1165円・税35円）
PC-PAGE 8　東京パソコンマップ1990
A5判　定価1200円（本体1165円・税35円）
PC-PAGE 9　MS-DOS3.3全ファイル解説
A5判　定価1200円（本体1165円・税35円）
PC-PAGE 10　パソコンを思想する
A5判　定価1200円（本体1165円・税35円）
PC-PAGE 11　MS-DOSバッチファイル
A5判　定価1500円（本体1456円・税44円）
PC-PAGE 12　プロフェッショナルのパソコン利用術
A5判　定価1200円（本体1165円・税35円）
PC-PAGE 13　PDS白書
A5判　定価1500円（本体1456円・税44円）

翔泳社　　書籍案内

●トレーニングブック

花子トレーニングブック
SE編集部編　B5判変型　定価2060円（本体2000円・税60円）
日本語EXCELトレーニングブック
森なおみ著　B5判変型　定価2266円（本体2200円・税66円）
新松トレーニングブック
真堂彬著　B5判変型　定価2266円（本体2200円・税66円）
シルエットトレーニングブック
SE編集部編　B5判変型　定価1648円（本体1600円・税48円）
Ninja3トレーニングブック
真堂彬著　B5判変型　定価1800円（本体1748円・税52円）
一太郎Ver4トレーニングブック
SE編集部編　B5判変型　定価1600円（本体1553円・税47円）
一太郎Ver3トレーニングブック
SE編集部編　B5判変型　定価1600円（本体1553円・税47円）
花子Ver.2トレーニングブック
SE編集部編　B5判変型　定価1600円（本体1553円・税47円）
マルチプランVer.4.1トレーニングブック
山口博幸著　B5判変型　定価2000円（本体1942円・税58円）

●入門・ガイドブック

ハードディスク・クックブック
AP-Labo編著　A5判　定価2266円（本体2200円・税66円）
ビギン・ザ・98
田口みすず著　A5判　定価1545円（本体1500円・税45円）
ワープロユーザーのためのMIFES-98
清成啓次著　A5判　定価1545円（本体1500円・税45円）
パソコンLAN読本
杉山育央著　A5判　定価1600円（本体1553円・税47円）
今夜わかるMS-DOS
SE編集部編　A5判　定価886円（本体860円・税26円）
ハードディスクの使い方
江阪俊哉著　A5判　定価1000円（本体971円・税29円）
CD-ROMの冒険
菱川英一著　A5判　定価1600円（本体1553円・税47円）
アシストカードガイドブック
真堂彬著　A5判　定価1200円（本体1165円・税35円）
UNIX入門
日本電子計算株式会社編　A5判　定価1800円（本体1748円・税52円）
NEXT GUIDE BOOK
SE編集部編　B5判変型　定価1500円（本体1456円・税44円）
ワークス・ノート
SE編集部編　A5判　定価1500円（本体1456円・税44円）

●Microsoft PRESS

THE IBM PC & PS/2プログラマーズガイド
Peter Norton・Richard Wilton著　A5判　定価5200円（本体5049円・税151円）
PC & PS/2ビデオシステムプログラマーズガイド
Richard Wilton著　A5判　定価6000円（本体5825円・税175円）
プログラマのためのPCソースブック
Thom Hogan著　B5判　定価5200円（本体5049円・税151円）

●プログラミング

日本語MS-Windowsプログラマーズガイド
AP-Labo編著　A5判　定価2266円（本体2200円・税66円）
MS-Windowsアプリケーションスタイルガイド
AP-Labo編　B5判　定価2884円（本体2800円・税84円）
Cビギナーズテキスト
松尾有祐著　A5判　定価1854円（本体1800円・税54円）
MS-C Ver4.0 クイックリファレンスガイド
SE編集部編　A5判　定価1854円（本体1800円・税54円）
dBASEⅢPLUSクイックリファレンスガイド
鈴木良美・堀江裕編　A5判　定価1854円（本体1800円・税54円）
日本語プログラミング言語Mind
片桐明著　A5判　定価2884円（本体2800円・税84円）
dBASEⅢPLUSプログラミングガイド
鈴木良美著　A5判　定価2060円（本体2000円・税60円）
80387活用プログラムコレクション
安田信行著　B5判　定価2884円（本体2800円・税84円）
Quick BASIC Ver4.2ダイジェストガイド
SE編集部編　A5判　定価1648円（本体1600円・税48円）
TURBO C Ver2.0ダイジェストガイド
塚本吉彦編著　A5判　定価1648円（本体1600円・税48円）
Quick Cプログラミングブック
塚本吉彦著　B5判変型　定価1800円（本体1748円・税52円）
MS-C Ver5.1オンラインリファレンス
谷口崇著　A5判　定価2200円（本体2136円・税64円）
日本語でプログラムを書く方法
加野島英渡・大網呼人著　A5判　定価1200円（本体1165円・税35円）
Turbo C2.0プログラミングブック
田中正造著　B5判変型　定価2000円（本体1942円・税58円）
QuickBASIC Ver4.5プログラミングブック
塚本吉彦著　B5判変型　定価2000円（本体1942円・税58円）
Quick C Ver.2.0ランタイムライブラリリファレンス
マイクロソフト株式会社監修
A5判　定価2400円（本体2330円・税70円）
QuickBasic Ver.4.5ステートメント&関数リファレンス
マイクロソフト株式会社監修
A5判　定価2400円（本体2330円・税70円）
QuickBASIC スタイルブック
イント・ツーワン著　B5判変型　定価2300円（本体2233円・税67円）

●エッセイ

読書するプログラマ
祐安重夫著　A5判　定価1648円（本体1600円・税48円）
デジタル版徒然草
園田康博著　B6判　定価1545円（本体1500円・税45円）
15センチな恋人関係
須藤慎一・東千峰子著　四六判変型　定価980円（本体951円・税29円）

●AXパソコン関連書籍

AXテクニカルレポート
A4判変型　定価3296円（本体3200円・税96円）
AX QUARTLY REPORT No.1
A4判変型　定価2575円（本体2500円・税75円）
AX QUARTLY REPORT No.2
A4判変型　定価2575円（本体2500円・税75円）
AX QUARTLY REPORT No.3
A4判変型　定価2575円（本体2500円・税75円）
AX QUARTLY REPORT No.4
A4判変型　定価1500円（本体1457円・税43円）
AX QUARTLY REPORT No.5
A4判変型　定価1500円（本体1456円・税44円）
AX QUARTLY REPORT No.6
A4判変型　定価1500円（本体1456円・税44円）
AX QUARTLY REPORT No.7
A4判変型　定価1500円（本体1456円・税44円）
AX QUARTLY REPORT No.8
B5判　定価1200円（本体1165円・税35円）

●通信・ネットワーク

PDS活用ガイド
デジタル・コミュニケーション・ラボ著　A5判　定価1854円（本体1800円・税54円）
画像通信ガイド
矢澤宏行著　A5判　定価1854円（本体1800円・税54円）
BBS開局ガイド
デジタル・コミュニケーション・ラボ著　A5判　定価1854円（本体1800円・税54円）
海外ネットワークアクセスガイド
海外ネットワーク研究会編著　B6判　定価1500円（本体1457円・税43円）
オンラインソフトウェアレビュー
堺誠著　A5判　定価1800円（本体1748円・税52円）
ネットワークINアメリカ
知野明著　A5判　定価1200円（本体1165円・税35円）
PC-VANのすべて
知野明著　A5判　定価1200円（本体1165円・税35円）
ビジネスマンのためのベータベース活用ガイド
鈴木尚志著　A5判　定価1500円（本体1457円・税43円）

翔泳社の本は全国のどの書店でもお求めになれます．店頭にない場合は，「至急」とお申し出のうえ，ご注文いただければ10日位でお届けできます．

株式会社　翔泳社
東京都千代田区平河町2-4-14平河町KSビル4F　〒102
TEL 03-3263-0447　FAX 03-3263-4526　担当：吉田　種村

●イベント

■1月30日(水)~2月1日(金)

page '91
「インテグレーテッド・ページネーション」をテーマに、エレクトロニック・プリプレスシステム、コンピュータ・パブリッシングシステムを展示。
会場：池袋サンシャインシティコンベンションセンターTOKYO／入場料金：500円
問い合わせ先：(社)日本印刷技術協会　TEL 03-384-3111

■2月14日(木)~2月16日(土)

MACWORLD Expo/TOKYO '91
Macintoshの関連製品を一堂に展示。日本初の開催。
会場：幕張日本コンベンションセンター　入場料金：2,000円
問い合わせ先：日刊工業新聞社事業部　TEL 03-3292-3561

■1月23日(木)~1月26日(土)

第6回ADEEジャパン'91
IC設計のための最新技術、機器を展示。
第8回エレクトロテスト・ジャパン'91
検査・試験装置の最新技術、機器を展示。
第20回インターネプコン・ジャパン'91
プリント回路基盤作成のための最新技術・関連機器を展示。
第11回国際レーザー／アプリケーション'91
理化学・産業分野のレーザー関連製品、技術を展示。
会場：幕張日本コンベンションセンター／入場料金：3,000円
問い合わせ先：カーナーズエクスポジションジャパン(株)　TEL 03-349-8501

■2月27日(水)~3月1日(金)

ボード・コンピュータ'91
CPU、メモリ、周辺機能ボードなどの産業用ボード・コンピュータを一堂に展示。'91スイッチング電源システム展、EMC・ノイズ対策技術展、'91小形モータ技術展も同時開催。
会場：池袋サンシャインシティコンベンションセンターTOKYO／入場料金：1,000円
問い合わせ先：日本能率協会　TEL 03-3434-1377

information・page 1

Microsoft University

月12日(火)~2月15日(金)

Microsoft LAN Manager Programming Environment
LANマネージャ用アプリケーション開発のための入門コース。
会場：富士ソフトウェアMUセミナールーム
受講料：150,000円(4日間)

■2月25日(月)~3月1日(金)

Microsoft LAN Manager Applications Programming
LANマネージャ用アプリケーション開発のための上級コース。
会場：富士ソフトウェアMUセミナールーム
受講料：180,000円(5日間)

■2月26日(火)~3月1日(金)

MS OS/2 Programming Environment
OS/2アプリケーション開発のための基本コース。
会場：インテルジャパン東京トレーニングセンター・MUセミナールーム
受講料：120,000円(4日間)

■3月4日(月)~3月8日(金)

MS OS/2 Applications Programming
OS/2のAPIを使った実習中心のコース。
会場：富士ソフトウェアMUセミナールーム
受講料：150,000円(5日間)

MS OS/2 Presentation Manager Programming Environment
OS/2 PMアプリケーション開発のための入門コース。
会場：インテルジャパン東京トレーニングセンター
受講料：150,000円(5日間)

Microsoft Universityは、OS/2、Windowsアプリケーション開発者のために定期的に開催されているセミナーです。
問い合わせ先：富士ソフトウェア(株)MU教育センター　TEL 03-3452-

●セミナー

■2月26日(火)~2月27日(水)

Macintoshの学校「マルチメディア実践トレーニング」
レーザーディスクとハイパーカードを使ったマルチメディアトレーニング
会場：池袋西武百貨店SMA館
受講料：109,180円(2日間)テキスト代・機材使用料込
問い合わせ先：西武百貨店IDFプロジェクト室Macintoshの学校
TEL 03-5992-8890

NEWS & NEW PRODUCTS

1 日本IBM，必要メモリを抑えたOS/2 1.3を発表 SEは2MB，EEは3MBで動作

日本アイ・ビー・エム（株）（TEL 03-3586-1111）は，OS/2の最新版「IBMオペレーティング・システム/2（OS/2）1.3」を発表した．

「OS/2 Standard Edition 1.3」は必要メモリ容量を従来の2.5MBから2MBに，「OS/2 Extended Edition 1.3」は3.5MBから3MBにそれぞれ抑え，さらに必要ハードディスク容量も削減した．フォントの充実や，コミュニケーション・マネージャ（Extended Edition）の強化による各種通信システムサービスの追加なども図られている．

同時に発表されたサーバー・プログラム「OS/2 LAN SERVER 1.3」（183,100円）と合わせて利用することで，トークンリング上だけでなく，イーサネット上のDOS端末も取り込んだLANを構築できる．

■おもな仕様

対応機種：PS/2シリーズ，既存のOS/2対応モデル／価格：54,400円（OS/2 Standard Edition 1.3），132,800円（OS/2 Extended Edition 1.3）

2 日本電気，16MHzのV30を搭載した98NOTEの後継機など PC-9800シリーズの新モデル5種を発売

日本電気（株），日本電気ホームエレクトロニクス（株）（NECパソコンインフォメーションセンター：TEL 03-3452-8000）は，PC-9800シリーズの新製品として，高速処理が可能な16ビットノート型パソコンと拡張性や操作性に優れた普及型のデスクトップパソコン４機種を発売した．

「PC-9801NV」（愛称：98NOTE NV，写真１）は，同社のノート型パソコンPC-9801N（愛称：98NOTE）の後継機に位置づけられる機種で，CPUに16ビットマイクロプロセッサV30HL（16MHz，ノーウェイト）を採用しているため，処理速度が現行機に比べて約1.6倍向上している．また，標準で3.8時間のバッテリー駆動が可能で，オプションの大容量バッテリーパックを利用することにより，最長5.7時間の連続使用が可能．電源投入時に，電源切断直前の状態をただちに再現できるレジューム機能もある．

デスクトップタイプの「PC-9801DX2」（写真２左），「同5」，「同/U2」（写真２右），「同/U5」は，５インチフロッピーディスク搭載型ディスクトップ機の主力機種であるPC-9801RX21，同51，ならびに3.5インチフロッピーディスク搭載型のデスクトップ機の主力機種であるPC-9801EX2，同EX4のそれぞれ後継機に位置づけられる製品で，メモリ専用スロットに12MBのメモリが増設可能で，拡張スロットを使用することにより最大14.6MBまでメモリが搭載できる．さらにフロッピーディスクモデルである「DX2」「DX/U2」では，最大100MBの固定ディスクを内蔵できる．また，周辺機器構成や動作条件などの利用環境を設定するディップスイッチの機能がメニュー画面にしたがって設定できる．グラフBIOSを高速化しているため，グラフ描画速度が現行機に比べて1.2〜1.6倍向上している．

■PC-9801NV

プロセッサ：V30HL（16MHz）／メモリ：640KB／

写真１

ワンラインNEWS
日立マクセルはフォーマット済みフロッピーディスクを発売した．フォーマットは日本電気のPC-9800シリーズに対応している．

NEWS & NEW PRODUCTS

外部記憶装置：3.5インチFDD×1／拡張スロット：16ビット用×1／価格：248,000円

■PC-9801DX2

プロセッサ：80286（12MHz）／メモリ：640KB／外部記憶装置：5インチFDD×2／拡張スロット：16ビット用×4／価格：318,000円

■PC-9801DX5

プロセッサ：80286（12MHz）／メモリ：640KB／外部記憶装置：5インチFDD×2・40MB HDD／拡張スロット：16ビット用×4／価格：468,000円

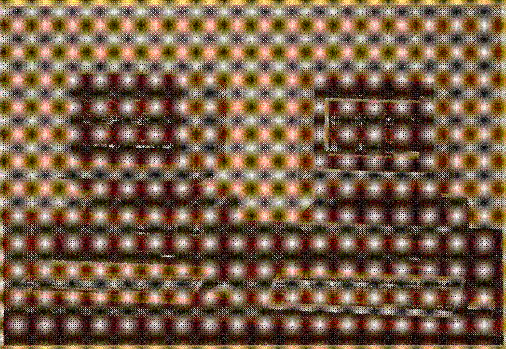
■PC-9801DX/U2

プロセッサ：80286（12MHz）／メモリ：640KB／外部記憶装置：3.5インチFDD×2／拡張スロット：16ビット用×4／価格：318,000円

■PC-9801DX/U5

プロセッサ：80286（12MHz）／メモリ：640KB／外部記憶装置：3.5インチFDD×2・40MB HDD／拡張スロット：16ビット用×4／価格：468,000円

写真2



3 富士通，486を搭載したFM Rシリーズの最上位機種を発売

富士通（株）（TEL 03-3216-3211）は，80486を搭載し，内蔵光磁気ディスクユニット，内蔵ICメモリカードリーダライタなどのオプションユニットの内蔵化を実現した，FM Rシリーズの最上位機種「FM R-80HL1」を発売した．FM R-80HL1は，80486の搭載に加え，CPUに内蔵の8KB，外部に32KBのキャッシュメモリを標準装備し，同社のFM R-70HX3に比べて約1.9倍の処理速度を実現している．グラフィック描画機能は，高速グラフィック・アクセラレータ「HSGAII」の搭載により，70HX3に比べ約2～6倍の処理速度を実現している．

また，拡張性も重視され，補助記憶装置として提供される，3.5インチ128MBの「内蔵光磁気ディスクユニット」（91年第3四半期発売予定），ノートブックパソコン（FM R-50NB1など）とのデータのやり取りを行う「内蔵ICメモリカードライタ」も本体に内蔵することができる．

5インチ，3.5インチの基本モデルにそれぞれハードディスクの容量により，4モデルがある．

■FM R-80HL1D7

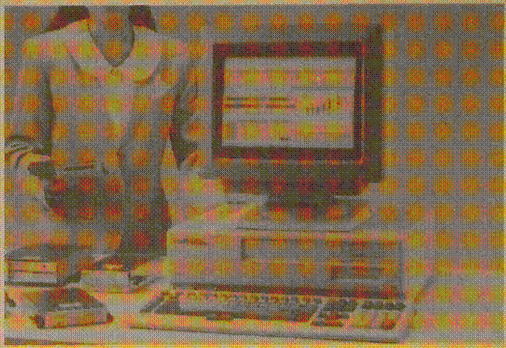
プロセッサ：80486（25MHz），内蔵8KB+外部32KBキャッシュメモリ／メモリ：16KB／外部記憶装置：5インチFDD×2・170MB HDD／拡張スロット：4／価格：1,920,000円

■FM R-80HLB8

プロセッサ：80486（25MHz），内蔵8KB+外部32KBキャッシュメモリ／メモリ：16KB／外部記憶装置：5インチFDD×2・85MB HDD／拡張スロット：4／価格：1,720,000円

■FM R-80HL1C7

プロセッサ：80486（25MHz），内蔵8KB+外部32



NEWS & NEW PRODUCTS

KBキャッシュメモリ／メモリ：16KB／外部記憶装置：3.5インチFDD×2・170MB HDD／拡張スロット：4／価格：1,920,000円

■FM R-80HL1A8

プロセッサ：80486（25MHz），内蔵8KB+外部32KBキャッシュメモリ／メモリ：16KB／外部記憶装置：3.5インチFDD×2・85MB HDD／拡張スロット：4／価格：1,720,000円

4 ソニー，NEWSの新機種を発売 ラップトップWSでは最高速の17MIPSを実現

ソニー(株)（NEWSインフォメーションセンター：TEL 03-3448-3553)は，NEWSの新機種として，CPUにRISCプロセッサを搭載し，ラップトップ型のワークステーションとしては業界最高速の処理速度17MIPSを実現した「NWS-3260」を発売した．

「NWS-3260」は，NWS-1200シリーズのコンパクトサイズ，機動性を継承しながら，大幅な処理速度の向上，大容量化を図った．CPUにMIPS社のR3000(20MHz)，コプロセッサにR3010を搭載し，整数演算で17MIPS，浮動小数点で1.8MFLOPS（倍精度），11.6 SPECmarkの高速処理を実現している．メインメモリに16MBバイトを実装し，高速大容量406MBのハードディスクを搭載したほか，オペレーティングシステム「NEWS-OS 4.0」を標準で装備している．バックライト付き高解像度STN液晶ディスプレイ（1120ドット×780ドット)を採用し，別売のモデムボードも用意されている．

■おもな仕様

プロセッサ：R3000（20MHz）／メモリ：16MB／外部記憶装置：3.5インチFDD×1・406MB HDD／価格：1,750,000円

5 東芝，DynaBook386シリーズと TFTカラー液晶のラップトップ型パソコンを発売

(株) 東芝（TEL 03-3457-2100）は，CPUに32ビット80386SX（16MHz）を搭載した「DynaBook386 J-3100SXシリーズ」3モデルとカラー液晶のラップトップ型パソコン「J-3100SGX 101T」を発売した．

「DynaBook386」（写真）は，同社のJ-3100SSに比べて約3倍の処理能力を備えた32ビットブック型パソコンである．

また，同時に発表された「J-3100SGX 101T」は，CPUに32ビット80386（20MHz）を採用し，4,096色から16色を同時に表示できるTFT方式カラー液晶ディスプレイを搭載したカラーラップトップ型パソコンである．

■DynaBook386 J-3100SX 001モデル

プロセッサ：80386SX（16MHz）／メモリ：2MB

ワンラインNEWS

富士通は電話・ファクシミリ・コンピュータなどを扱う各情報を異機種端末間でやりとりできる通信情報処理装置「HMCP」を発売した．この装置は業界初だという．

NEWS & NEW PRODUCTS

／外部記憶装置：3.5インチFDD×1／拡張スロット：拡張インターフェース／価格：258,000円

■DynaBook386 J3100SX 002モデル

プロセッサ：80386SX（16MHz）／メモリ：2MB／外部記憶装置：3.5インチFDD×2／拡張スロット：拡張インターフェース／価格：298,000円

■J3100SX 021モデル

プロセッサ：80386SX（16MHz）／メモリ：1MB／外部記憶装置：3.5インチFDD×1・20MB HDD／拡張スロット：拡張インターフェース／価格：368,000円

■J3100 SGX 101Tモデル

プロセッサ：80386（20MHz）／メモリ：2MB／外部記憶装置：3.5インチFDD×1・100MB HDD／拡張スロット：2／価格：1,680,000円

6 日本IBM，光磁気ディスクが内蔵可能なPS/55Zの新モデルを発売

日本アイ・ビー・エム（株）（03-3586-1111）は，32ビット・パーソナルシステム「PS/55Z」シリーズに「モデル5530―S12」および「モデル5530-S14」（写真）の新モデル2種を発売した．

「モデル5530―S12」は，CPUに標準で2MB，システムボード上で6MBまでのメモリを拡張することができる．また，オプションで40MBのハードディスクを搭載することができる．

「モデル5530―S14」は，オプションで，127MBの3.5型光磁気ディスクを内蔵できる「3.5型光磁気ディスク駆動機構/A」を装備できる．光磁気ディスクは，従来の3.5インチFDDの80枚分のデータを1枚に収納する容量をもつ記憶メディアで，ディスク・カートリッジは読み書き自在の127MBと再生専用の122MBの2種類を利用することができる．さらにS14システム装置に「IBM DOSバージョン J4.0」をあらかじめ導入した「モデル5530―SJ4」も同時に発表された．

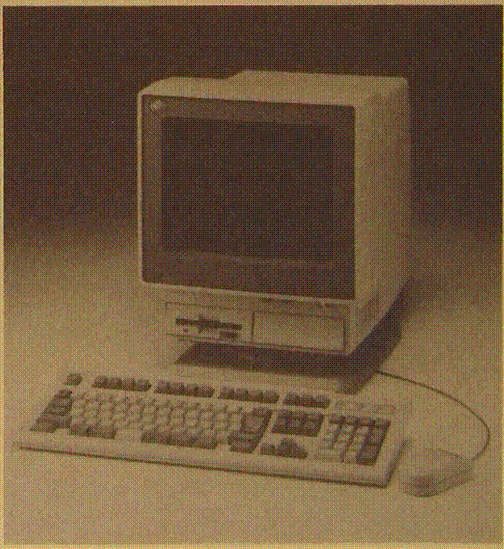
両機種に搭載された3.5インチドライブは，1.4MB，1.2MBフォーマットをサポートしている．

■モデル5530-S12

プロセッサ：80386SX（16MHz）／メモリ：2MB／外部記憶装置：3.5インチFDD×2／拡張スロット：16ビット用フルサイズ×2／価格：398,000円

■モデル5530-S14/SJ4

プロセッサ：80386SX（16MHz）／メモリ：2MB／外部記憶装置：3.5インチFDD×1・40MB HDD／拡張スロット：16ビット用フルサイズ×2／価格：560,000円（S14），588,000円（SJ4）



NEWS & NEW PRODUCTS

7 日本IBM，ハードディスクを拡張した5530Tの新モデル3種を発売

日本アイ・ビー・エム（株）（TEL 03-3586-1111）は，「IBMパーソナルシステム/55」の最下位機種でOS/2対応のディスクトップ型ワークステーション「PS/55モデル5530-T」のハードディスクを拡張した新モデルを発売した．

今回発表されたモデルは，T02／T04／T08の３機種で，「PS/55モデル5530-T04およびT08」（写真はT80）は，ハードディスク容量がこれまでの30MB，60MBから40MB，80MBに増加した．また「PS/55モデル5530-T02」は，3.5型ディスケットドライブを２台内蔵している．

■おもな仕様

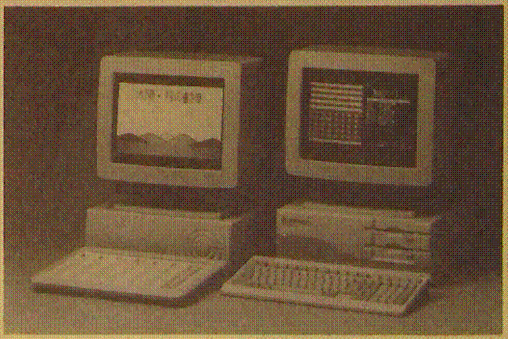
プロセッサ：80386（20MHz）／メモリ：2MB／外部記憶装置：3.5インチFDD×２（5530-T02），3.5インチ FDD×１・40MB HDD（5530-T04），3.5インチ FDD×１・80MB HDD（5503-T08）／拡張スロット：32/16ビット用フルサイズ×２／価格：602,000円（5530-T02），780,000円（5530-T04），960,000円（5530-T08）

8 日本電気，ネットワーク型教育システム PC SEMI Kシリーズを発売

日本電気（株）（TEL 03-3454-1111）は，学校，塾，予備校などの幅広い分野での利用が可能なネットワーク型教育システム「PC SEMI Kシリーズ」を発売した．

「PC SEMI Kシリーズ」は，先生と生徒のパソコンをローカルエリアネットワークで接続し，学習指導を行うことができる．パソコンの文字情報やグラフィックス情報を扱うことが可能な「PC SEMI KS」，VTRやテレビなどの映像情報や語学教材，視聴覚資料などの音声情報も扱うことができる「PC SEMI KA」の２モデルがある．LANには，32Mbpsの高速LANを採用し，一斉・グループ・個別という３種類のモードを利用して先生から生徒に教材を送ったり，生徒の学習状況を先生の手元で把握できるオートスキャンモニタ機能がある．生徒用端末として，最高80台までのパソコンを接続することができる．

価格は，KSが6,900,000円，KAが9,400,000円（先生用端末１台＋生徒用端末20台），ソフトウェア「PC ANGLE XE」が１教室あたり500,000円．



ワンラインNEWS
東芝は4MビットDRAMの量産拠点となる四日市工場を建設している．今後数年間で1,000億円の資金を投資し92年に稼働する予定．

NEWS & NEW PRODUCTS

9 富士通，教育分野向けに機能強化したFM TOWNS新モデル2種を発売

富士通(株)(TEL 03-3216-3211)は，ハイパーメディアパソコン「FM TOWNS」を教育分野向けに機能強化したFM TOWNSモデルS1/S2の後継機種として，「FM TOWNSモデルSH/SF」の2種を発売した．

また同時に，FM TOWNS用の教育ソフトの強化・充実を図り，発売を開始した．

■おもな仕様

プロセッサ：80386 (16MHz) ／メモリ：2MB／外部記憶装置：CD-ROM×1・3.5インチFDD×2 (モデルSF)，CD-ROM×1・3.5インチFDD×2・40MB HDD(モデルSH)／拡張スロット：ビデオカード用×1・拡張RAMモジュール用×3・387カード用×1・汎用バス対応スロット×2／価格：358,000円 (モデルSF)，508,000円 (モデルSH)

10 日本IBM，高解像度XGA対応のPS/2最上位モデルなどを発売

日本アイ・ビー・エム (株) (TEL 03-3586-1111) は，「IBMパーソナルシステム/2(PS/2)」(英文仕様)ではじめて32ビット80486を搭載し，処理能力を大幅に向上させた最上位床置型「PS/2 モデル95 XP 486」と，卓上型「PS/2 モデル90 XP 486」を発売した．

「PS/2 モデル95 XP 486」は，PS/2ファミリーの最上位機種で，CPUに32ビット80486 (33MHz) を搭載し，ハードディスクは標準で320MB，最大で1.6GBまで拡張することができる．現行のモデル80-A31と比べ，最大で2〜3.4倍の処理能力の向上を実現している．

「PS/2 モデル90 XP 486」は，PS/2ファミリーの高機能卓上型で，CPUに32ビット80486 (25MHz) を搭載し，ハードディスクは標準で160MB，最大で800MBまで拡張することができる．現行のモデル70-A21と比べ，商用計算で1.8倍，数値演算で約3倍の処理能力を実現しており，CADやAIなどのエンジニアリング・システムをオフィス環境で実現できる．

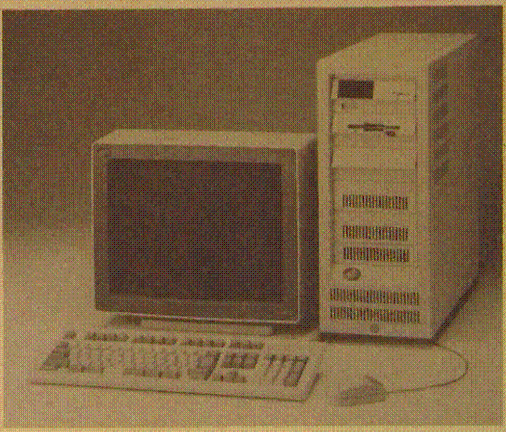
またPS/2関連製品として，高解像度のディスプレイ表示規格 (XGA:Extended Graphics Array) に対応したグラフィック・アダプター「XGA表示アダプター/A」(186,200円) や，PS/2をイーサネット・ネットワークに接続するための「イーサネット用ネットワーク・アダプター/A」(97,800円) なども同時発売した．

■モデル95 XP 486

プロセッサ：80486 (33MHz) 8KBキャッシュメモリ／メモリ：4MB／外部記憶装置：3.5インチFDD×1・320MB HDD／拡張スロット：32/16ビット用フルサイズ×6／価格：2,961,000円

■モデル90 XP 486

プロセッサ：80486 (25MHz) 8KBキャッシュメモリ／メモリ：4MB／外部記憶装置：3.5インチFDD×1・160MB HDD／拡張スロット：32/16ビット用フルサイズ×2・16ビット用×2／価格：2,171,000円



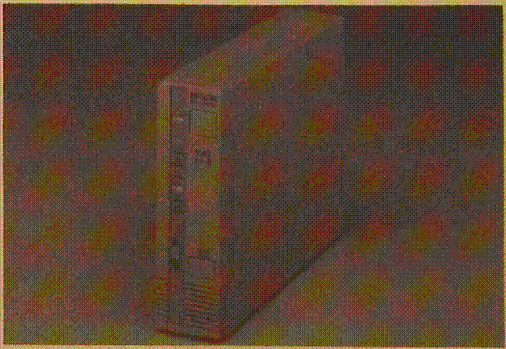
沖電気はワープロなど熱転写式プリンタで繰返し利用できるリサイクル型の感熱記録材料を開発した．これは温度によって白くなったりするプラスチックフイルム．

NEWS & NEW PRODUCTS

11 エプソン，V.42・MNPクラス5搭載のパーソナルモデムを発売

セイコーエプソン(株)，エプソン販売(株)（TEL 03-3377-3500）は，通信中のエラーフリー機能としてCCITT（国際電信電話諮問委員会）V.42およびMNPクラス5を標準搭載した2400bps対応のパーソナルモデム「エプソンSR-240V」を発売した．

「エプソンSR-240S」の上位互換となっており，既存の市販アプリケーションソフトをそのまま使用することができる．価格は，43,800円．



12 日本電気，98NOTEなどに対応の4MB，8MB用増設メモリを発売

日本電気（株），日本電気ホームエレクトロニクス（株）（NECパソコンインフォメーションセンター：TEL 03-3452-8000）は，PC-9800シリーズ用の4MBと8MBの増設RAMカードを発売した．

■PC-9801N-03（4MB），PC-9801N-04（8MB）
対応機種：PC-9801NV，PC-9801NS，PC-9801N，PC-DO+／価格：148,000円（PC-9801N-03），278,000円（同04）／出荷時期：1991年1月下旬

13 アイ・オー・データ機器，55Z用SIMメモリなどを発売 増設メモリのラインナップを強化

（株）アイ・オー・データ機器（0762-21-4812）は，各種メーカー用の増設メモリのラインナップ強化を図るため，以下の新機種の発売を開始した．

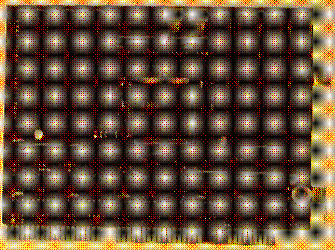
■IBM PS/55Z モデル5530Z SX用SIMモジュール「PS-SIM55-2M」
価格：38,000円

■東芝DynaBooK用増設RAMカード「PIO-31SS-4MS/8MS」
価格：75,000円（PIO-31SS-4MS，4MB），145,000円（PIO-31SS-8MS，8MB）

■エプソンPC-386M専用RAMボード「EP-386M-2M」
価格：45,000円

■IBM PC/AT，AX用EMSプラス増設RAMボード「PIO-AX34F-8M/16M」（写真）
価格：168,000円（8MB用），295,000円（16MB用）

■NEC PC-9801シリーズ用大容量メモリ32MB搭載I・Oバンク方式RAMボード「PIO-9234-32M」
価格：590,000円



ワンラインNEWS
コンピュータ専門商社のオービックはSI事業の主力製品として官公庁や自治体革新を支援するシステム6種を開発した．今年4月から販売する．

NEWS & NEW PRODUCTS

14 メルコ，DRAMを用いたシリコンディスクを発売

（株）メルコ（TEL 052-251-6891）は，シリコンディスク「SDAシリーズ」を発売した。

シリコンディスク「SDAシリーズ」はDRAMを記憶媒体に用いた機械的な構造をもたない純電子式の次世代外部記憶装置で，高速で高信頼という特性をもつ。

■おもな仕様

対応機種：PC-9800，FC-9800，PC-286/386シリーズ対応／価格：79,800円（4MB，SDA-4000），140,000円（8MB，SDA-8000），200,000円（12MB，SDA-12M），320,000円（20MB，SDA-20M）※すべてコントロールユニットBP-70とセット。

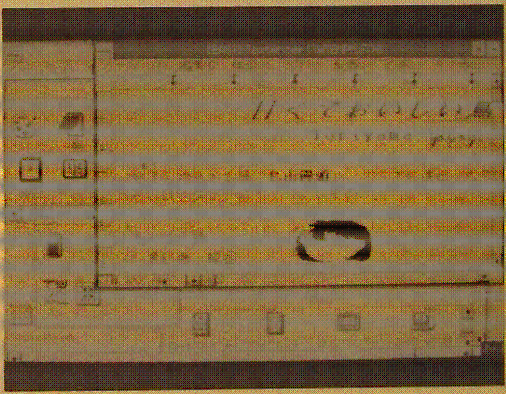
15 日本IBM，ワークステーション用最高速プリンタを発売

日本アイ・ビー・エム（株）（TEL 03-3586-1111）は，PS/55用の卓上型プリンタとして，ワイヤー・ドット・シリアル方式では業界最高速の印字速度をもつ「IBM5579-H02印刷装置」を発売した。

これは，印字ヘッドに48ピン・ヘッドを採用して，通常印字モードで毎秒200字，高速印字モードで毎秒漢字400字という印字速度を実現しており，現行モデルのIBM5577-H02印刷装置と比べ，65%以上速くなっている。価格は，818,000円。

16 イースト，Windows 3.0対応の日本語ワープロソフトを発表 オンラインヘルプやカラーアイコンに対応

イースト（株）（TEL 03-3374-1980）は，日本語MS-Windows 3.0上で稼働する日本語ワープロソフト「TextWriterバージョン2」を4月より発売する。Windows 3.0の特徴である，オンラインヘルプやカラーアイコンをサポートし，機種に依存しない日本語入力ドライバー「W1Win」を標準添付している。また，前バージョンで要望の多かったカーソルやスクロールスピードの問題も，高速入力モードを追加し，対応している。また，既存ユーザーには無償バージョンアップを行う。価格は，58,000円。



17 日本電気，日本語OpenDeskなどPC-9800用UNIX関連ソフトを販売開始

日本電気（株），日本電気ホームエレクトロニクス（株）（NECパソコンインフォメーションセンター：TEL 03-3452-8000）は，PC-9800シリーズの新製品の発売に合わせて，オペレーティングシステム「日本語OpenDesktop」を日本ではじめて製品化し，販売を開始した。

「日本語OpenDesktop」は，オペレーティングシステムUNIX System V リリース3.2をもとに，アイコン表示やマウスの利用で容易な操作を実現するグラフィカルユーザーインターフェースやネットワーク機能などを統合したオペレーティングシステムで，グラフィカルユーザーインターフェースとして，X Win-

NEWS & NEW PRODUCTS

dow上で利用するOSF/Motif, X.Desktopの日本語版を装備しているため，ビジュアルで容易な操作が可能．基本セットにはネットワーク機能として，通信プロトコルTCP/IPを利用した仮想端末機能やファイル転送機能，他のUNIXシステムをサーバーとしてファイルの共有ができるNFSクライアント機能，日本語MS OS/2を利用したシステムをサーバーとしてファイルやプリンタの共有ができるLM/Xクライアント機能があり，ネットワークの構築が容易である．MS-DOS実行環境機能により，日本語MS-DOS用のアプリケーションソフトの実行が可能．

また，日本語MS-DOS，日本語MS OS/2，日本語OpenDesktopシステムでファイルやプリンタを共有するためのソフトウェアLM/Xサーバを日本ではじめて製品化し販売を開始した．

■PC-UX/V(Rel3.2)日本語OpenDesktop基本セット
　価格：200,000円／出荷時期1991年2月下旬
■PC-UX/V(Rel3.2)日本語OpenDesktopサーバ
　価格：200,000円／出荷時期1991年2月下旬
■PC-UX/V(Rel3.2)日本語OpenDesktop開発セット
　価格：300,000円／出荷時期1991年2月下旬
■PC-UX/V(Rel3.2)LM/Xサーバ
　価格：350,000円／出荷時期1991年4月下旬

18 マイクロソフト，Microsoft WorksのDynaBook版を発売

マイクロソフト（株）（TEL 03-3363-1201）は，東芝ダイナブック対応版「Microsoft Works Version 2.1（東芝DynaBookシリーズ用）」を発売した．

「ワークス」は，ワープロ，表計算，データベース，通信の4つの機能をもつ統合ソフト．価格は，40,000円．

19 ツァイト，Z'S STAFF KiD98 Ver.3.0を発売

（株）ツァイト（TEL 03-3299-0463）は，PC-9801シリーズ対応のグラフィックソフト「Z'S STAFF KiD98 Ver.3.0」を発売した．エッジ・シェード・モザイク機能，16種の縦／横グラデーション機能，筆ペン機能など，多種のグラフィック機能をもつ．（株）アスキーから販売される．価格は，28,000円．

20 マイクロソフトとキヤノン販売がMacintosh用ソフトを共同開発

マイクロソフト（株）とキヤノン販売（株）（アップル販売企画部：TEL 03-3455-9131）は，アップル社製パーソナルコンピュータMacintosh用の電子メールソフト「Microsoft Mail 2.0日本語版」を共同開発し，販売を開始した．

「Microsoft Mail 2.0日本語版」は，マッキントッシュのネットワーク・コンピューティング環境を利用し，クライアント／サーバ型のアーキテクチャを採用したオープン・システム指向の電子メールシステムである．価格は，「Microsoft Mail 2.0サーバ・パック日本語版」が78,000円，「Microsoft Mail 2.0ワークステーション・パック日本語版」が278,000円．

21 アスキー，インフォミックス社と提携強化

（株）アスキー（TEL 03-3797-6506）は，米国インフォミックス社(Informix Corporation，本社：米国カリファオルニア州）の新株式671,833株（総額6,718,330ドル)を取得した．これはインフォミックス社の発行する株式のうち5％強にあたり，両者の関係は資本・業務の両面でいっそう強固になる．

アスキーは，インフォミックス社と1983年にリレーション・データベース・ソフト「Informix」の日本語化および販売で，1988年にアプリケーション開発言語搭載の統合型表計算ソフト「Wingz」の日本語化および販売で，それぞれ業務提携を結んでおり，日本国内でインフォミックス社製品を独占販売してきた．

ワンラインNEWS
三菱電機はワンチップマイコンの生産を拡充する．月産2300万個の規模を今年3月をメドに2500万個とし，91年度中には業界トップの3000万個に引き上げる．

NEWS & NEW PRODUCTS

22 マイクロソフト，片貝システム研究所とジョイントし公認トレーナー制度「MOT」を開始

マイクロソフト（株）は，（株）片貝システム研究所をジョイントパートナーとして，自社製品の公認トレーナー制度「MOT」を1991年2月から開始する．

「MOT」は，試験により優秀と認められたインストラクターに贈られる資格であるが，必ず事前の講習を受け，操作能力および指導力の収得を必須条件としている．

当初開設するのは，統合型計算プログラム「Microsoft Excel」のトレーナーのコースで，東京をはじめ全国4都市で2か月に1回講習会（2日間）を開催する．問い合わせは，（株）片貝システム研究所内MOT事務局（03-3797-3868）まで．

23 アスキーネットPCS，9600bpsの高速通信サービス開始従来より8倍高速な情報サービスを提供

（株）アスキー（TEL 03-3486-7111）は，パソコン通信ネットワークサービス・アスキーネット「PCS」で，9,600bpsの高速モデムが利用可能なアクセスポイントを開設し，12月12日よりサービスを開始した．9,600bpsの高速パソコン通信はこれまで小規模なネットワークなどではみられたが，本格的な商用ネットワークでははじめてのことだ．これにより，最高で従来の8倍の速度で通信することが可能になる．

今回開設したアクセスポイントは，CCITTの制定・勧告によるV.32（通信速度規格），V.42（エラー訂正規格），V.42bis（データ圧縮規格）に準拠している．この規格に準拠したモデムでPCSに接続すると，データ圧縮の効果などにより最高19,200bpsの実効通信速度を得ることができる．これは，従来の2,400bpsにくらべて8倍の速度にあたる．通信速度の高速化により，画面表示が速くなるのをはじめ，ファイル転送の時間なども短縮される．

24 OS-9000の開発パッケージがVer.1.1にバージョンアップオプショナルパッケージも提供開始

マイクロウェア・システムズ（株）（TEL 03-3257-9000）は，昨年7月より販売しているOS-9000 PCデベロップメント・パッケージ（対応機種：PC-9800シリーズ，FM R，AX）をVer.1.1にバージョンアップするとともに，OS-9000の新しい開発ツールの発売を開始した．

「OS-9000 PC Development Package Ver.1.1」（190,000円）は，SCSIのサポートをはじめグラフィック機能の追加，「PC-DOSファイルマネージャ(PCF)」によりDOSファイルの読み書きをシェルの標準I/Oレベルとして可能にするなど，従来のVer.1.0にくらべて大幅に機能アップしたもの．

また，新規に2種の開発用オプショナル・パッケージを提供する．ひとつは，OS-9000を自社製オリジナルハードウェアに移植するための移植ライセンスパッケージ「OS-9000 Port Pak」（600,000円），もうひとつはOS-9000上でインターネットをサポートするための開発パッケージ「OS-9000/ISP」（INT9KPK：500,000円/INT9KEI：120,000円）．

PCFで読み書き可能なDOSフォーマットは，MS-DOSの5インチ2HD，PC-DOSの5インチ2HD，AX用3.5インチ2HCとパーティション付きハードディスク（small/large FAT）．

東京エレクトロンは光通信用電子部品メーカーのデュポンオプトエレクトロニクスと販売代理店契約を締結し，デュポンオプトの全製品の販売権を取得した．

佐々木哲のニューヨーク

N.Y. システム通信

90年はATが世界を制覇

一歩日本の外に出ると，そこはIBM PC/AT互換機の世界です．生みの親であるIBMさえコントロールが効かなくなるほど広まったIBM互換機．まさに世界標準となっています．ソフトウェア，ハードウェアともに市場から好きなものが選択できる．この状況は米国をはじめ西欧諸国，アジアにもあてはまります．

例外もいくつかありました．ところが90年はまさに世界中でこの状況に．まず，国産機保護のため輸入規制していたブラジルが規制緩和しました．不況に喘ぐとはいえ南米最大の国ブラジル．規制にも関わらずPC，PC/AT，PS/2が「密輸入」されていたのが追認されたかたちです．

長い間の冷戦のためハイテク技術の輸出がCOCOMにより規制されていた共産圏．ソ連，東欧諸国でさえ，もう大丈夫です．東西緩和につれ386どころかINTEL 486CPUでさえ，COCOM規制がゆるめられました．大容量ハードディスク，WORMドライブなども規制外に．

現地ソ連では，1MBメモリ付きAT互換マザーボードが6,500ルーブル（約４万円）で手にはいるほどになってきたとか．どっかの国より安いですね．この巨大なマーケットを逃す手はないと，米国のソフトウェア会社もぞくぞくとモスクワに支社を設立しつつあります．現地で売れなくても，豊富なソフト技術者目当てのところもあります．

これらの国を含めヨーロッパからも，優れた，興味深いソフトウェアが現われつつあります．世界標準機の上で作られたシステムであれば，すぐに世界に向け輸出できます．まさにグローバル．

残るは非互換の最後の砦，日本です．これじゃソフトはやっぱり輸入超過ですね．

日本ではどうするの？

MS-Windows 3.0は，この春の発売から全世界でもう150万本以上販売されたとか．これはMicrosoftから直接エンドユーザー向けに販売された分だけです．ハードウェアバンドルなどOEM向けにはさらに100万本売れたとか．ソフトウェアも爆発的に充実してきています．今月はWindowsやOS/2ベースなどWYSIWYGのソフトのもつフォントという共通の悩みについて書きます．

ワードプロセッサをはじめ，表計算，データベースなどでもDTP的な表示，印字の美しい表現能力がWindowsの売り文句です．当然文字フォントは，サイズ，種類ともに豊富に使えるはずです．ところが，文字を拡大したら最後ギザギザでは困りもの．比較的高価なPostScriptプリンタなどを使うことはできます．しかしプリンタがさまざまなフォントをサポートしても，ディスプレイ上でそれを表示できなければWYSIWYGではありません．ディスプレイにも適切なフォントが必要です。フォントの供給とマネージメントが鍵です．

いままで，非Windowsベースのソフトではこの処理をそれぞれのソフトが別個に行っていました．いろいろなサイズのフォントがほしいときは，付属のフォントジェネレータでビットマップフォントを生成します．画面にWYSIWYG表示するには，同じ作業をディスプレイのためにします．汎用的なフォントジェネレータとして，BitstreamのFontwareなどはWordPerfectなどさまざまなソフトに添付されていました．

これでは，ユーザーにとっては手間がかかり面倒です。いくつものサイズでフォントをあらかじめ生成しておくとディスクスペースの10MBなどすぐになくなってしまいます．さらには，市場にはサードパーティのいろいろなソフト用のフォント，フォントジェネレータソフトがあふれ混乱しました．

MS-Windowsでは，ソフト側の負担はかなり軽減されました．Windows側でディスプレイ，プリンタなどのフォントを各ソフト共通に面倒みてくれます．しかし，ユーザーにとってはフォントを前もって用意する面倒さは同じです．

つい最近，ようやくこの状況が改善されてきました．表示，印字するときに，on-the-flyでフォントを生成してくれるフォントマネージャの出現です．

まずはPostScriptの総元締めAdobeからAdobe Type

ワンラインNEWS
日本電気はオフコン「NECシステム3100シリーズ」の下位機種として「同/X01」など４機種を発表した．HDDにキャッシュを内蔵しアクセス時間を短縮している．

Manager (ATM)*[1]が，またフォント資源を豊富に揃えているBitstreamからはFaceLift*[2]が発売されました．さらにSuperPrint，MoreFont 2.0などぞくぞくと続きます．問題なくお薦めできるのはATMとFaceLiftでしょう．フォントの美しさではATMの勝ち．OS/2版のATMはIBM OS/2 1.3にも標準添付されています．使いやすさ，メモリ使用量の少なさの点ではFaceLiftが勝ちます．

これらを使えば，好きなサイズのフォントを前もって準備することなくWindows上で使用できます．それぞれ使い勝手は異なりますが，原理的にはWindowsのGraphics Device Interface (GDI) とスクリーン，プリンタドライバの間に入りフォントをアウトラインのソースから生成します．好きなサイズのフォントを自動的にディスプレイ，各種プリンタに出力できます．Windowsのサポートするプリンタならドットプリンタだろうが出力可能．

それぞれのパッケージは13種類のタイプフェイスが付いてきます．ユーザーは必要に応じてタイプフェイスを追加購入していくわけです．

さて，また困ったことがひとつ．各マネージャのタイプフェイスのフォーマットがまた千差万別なのですね．ATMは当然AdobeのType 1フォントをサポートします．FaceLiftはBitstreamの新しいフォーマットSpeedoをサポートします．互換性はありません．

グラフィックフォーマットのように，このフォントを巡る混乱はまだまだ続きそう．遠くない将来には，MicrosoftとAppleからTrueTypeが参加しさらにひどくなるでしょう．でも，これは便利さという強いニーズを巡る争いですね．

ちなみに，このFaceLiftはWindowsを得意とする西独のGCAが製作したとか．

＊1:Type Manager for Windows $99.00,Adobe Systems Inc.
＊2:FaceLift $99.00,Bitstream Inc.,Cambridge,Catherine Hartley,phone: (617) 497-6222

マルチメディア用の新チップセット

ここにきてマルチメディアという言葉をよく聞くようになりました．プレゼンテーションに続く，大きな新分野ですね．

各所でMotion videoのデモを見る機会があります．たとえば，MS-Windows上で１つのWindowの中に外部ビデオソースの動画を表示できるような．要はVGAディスプレイ上でTVが好きなサイズで見れたり，その画面をキャプチャーしたりできるものです．

ハードウェアとして，VideoLogicのDVA-4000やIBMのM-Motion Videoが必要です．どちらも2,000ドル前後する高価なものです．

Chips & TechnologiesはNew Media Graphicsと共同で，この機能を持ったチップセットを開発しました．それを最初に使用した製品がNew Media GraphicsのSuper VideoWindows board*[3]です．価格は一気に695ドルと３分の１になりました．画像は，リアルタイムに24ビットでアプリケーションから自在にカット＆ペーストできます．M-Motion Videoにできることはすべてできるとか．残念ながら互換性はありません．同社ではこのボードのファミリーとして，TVチューナーボード（395ドル），画像，音声圧縮ボードも出しています．

このC&Tのチップセットを使った製品は数多く出てくることでしょう．ニューメディアもぐっと手ごろなものになりそうです．

＊3:Super VideoWindows board $695.00,New Media Graphics,780 Boston Rd,Billerica, MA 01821-5925,phone: (508) 663-0666 fax: (508) 663-6678

今月の何だこりゃ？

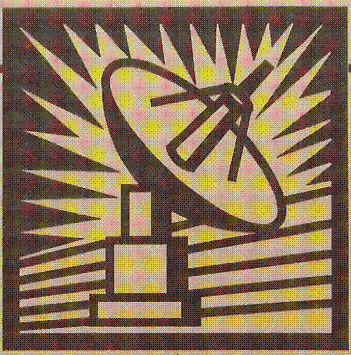
パソコンを使って作業をしているとき，ふと気がつくと同じキー操作を繰り返していることがよくあります．たとえば表計算ソフトで同じ値を各セルにつぎつぎと入力したり，エディタでつぎつぎと同じような修正を繰り返したり．せっかくコンピュータを使っているのに間抜けな話です．

Micro LogicはKey Watch*[4]というユーティリティソフトを発売しました．このソフトはメモリに常駐していて，ずっとキー入力を見張っています．いったんユーザーが同じキー入力の繰り返しを始めるとKey Watchはブザーを鳴らし，手伝いを申しでます．ユーザーはポップアップメニューを使い，一連のキー操作をホットキーに登録します．あとは簡単に同じ操作を繰返せるわけです．今までもキーマクロユーティリティは星の数程ありましたが，このように自己主張するのははじめて．さて，どこまで便利になるかな？それにしてもこのソフト，メモリ常駐量を3KBしか取らないのはりっぱです．

＊4:Key Watch $69.95,Micro Logic,P.O. Box 70 Dept. P,Hackensack, NJ 07602,phone: (800) 342-5930 or (201) 342-6518

Illustration : MASAMI NOZUE

Illustration : NOBUKI FUKUHARA



西川和久の

希望的観測

A4縦型ディスプレイのパソコンが欲しい！ GUIの普及で現実的な課題に……

1 なぜ10年たっても ディスプレイは変わらない

最近つくづく思うことがある．ディスプレイがついて，それを標準表示装置として使うパソコンができてからほぼ10年を経ている現在，未だに一般的な解像度が横80文字×縦25行というのは何かおかしいと．ユーザーは何も不自由を感じていないのだろうか!?

パソコンの使い方として，ワープロソフトの利用率が一番高いと聞いているが，ワープロで通常一番多いフォームはA4縦，すなわち縦66行（最大）ではないのだろうか．A4縦のフォームだと現在のディスプレイでは，たかだか1ページを見るのに画面を3回もスクロールする必要がある．はなはだ一覧性に欠けるこの画面サイズを，なぜ今でもメーカーが採用し，またユーザーも文句をいわないで使っているのか理解に苦しむ．もちろんワープロソフトだけではなく，表計算ソフトなども一覧性に欠けてしまうのはいうまでもない．ワープロ専用機やワークステーションを好んで使う人からよく聞くのは「画面が広い」という言葉である．パソコンの世界でも，なんとかならないのだろうか……．

最近，家電店へ行く機会があったので，パーソナルワープロ専用機をいろいろ眺めてきた．驚くことに20万円台（当然，実売価格はもっと安い）のマシンに，白黒であるがA4縦ディスプレイが搭載されているではないか！　いままで私がメーカーの方からA4縦型ディスプレイができないひとつ理由として聞いていたのは，「ディスプレイが高価になる」ということだった．ワープロの現状を見ると，それはいいわけとしか聞こえない．

2 マシンと心中…… 日本のパソコン文化

もうひとつの理由として，「アプリケーションが対応していないとA4縦ディスプレイが使用できない」という声もあった．もっともらしい理由であるが，米国のアプリケーションはCGA，EGA，VGA，SVGAなど，いろいろあるディスプレイ規格に容易に設定できるようになっている．アプリケーション側でディスプレイドライバを切り分けてパッケージに納めておき，ユーザー側で自分のシステムに合ったドライバを選択できる構成になっている．しかし，残念ながら日本ではこのような文化がなく（国産の代表的なマシンがビデオカードを交換できるカラクリになっていない），解像度を上げたければ上位モデルを買い直す必要があるし，アプリケーション側も複数のビデオカードに対応できる構成になっていない．

国産マシンを対象として，独自にビデオカードを作って販売している会社がないではない．しかし現状では，独自のビデオカードを作ったボードメーカーは，ビデオカードを持ち歩いて，「このビデオカードに対応してください」と有力アプリケーションを出している会

ワンラインNEWS
同じクロックなら実行速度は数パーセント速いといわれるエプソンのPCシリーズ．しかし内部拡張メモリが発売されていないためMemoryPRO386を使うと遅くなる．

社に頭を下げて回る必要がある．これではなかなか対応アプリケーションも増えず，アプリケーションが増えないからディスプレイボードを作らない……と，鶏と卵の関係に陥ってしまう．

これでは，当分の間A4縦ディスプレイをパソコンに接続することはできないと思ったが，ちょっと別の切り口で良い案が浮かんだ．

3 CUIからGUIへの流れのなかで

最近のパソコン界は，Windows 3.0やOS/2の登場でCUIの世界からGUIの世界に移ろうとしている．こうした環境が本格的に活用され出すにはまだ少し時間がかかると思われるが，これらの環境はその上で動くアプリケーションのプラットホームとなるため，たとえばWindows対応のアプリケーションであればマシンの種類・画面解像度を問わず，基本的にすべてのマシンのWindows上で作動可能となる．マシンの種類や接続している装置の問題は，Windowsシステムが吸収してくれる．A4縦型のディスプレイがあったなら，そのWindows用ドライバが必要なだけだ．

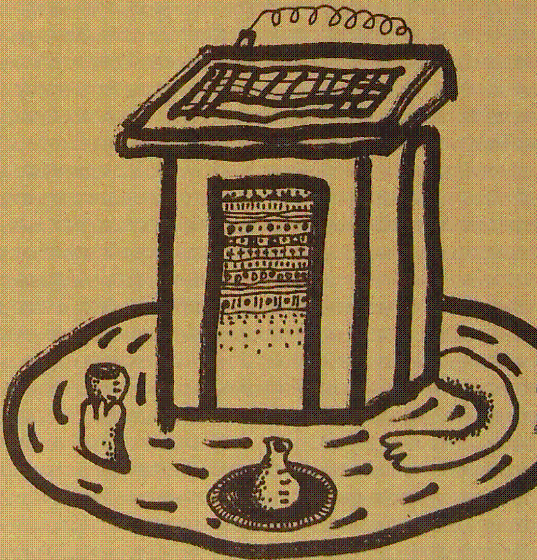
これなら先の「対応するアプリケーションがない」という問題もクリアされ，画面ドライバさえあれば（きちんと動くようにするにはかなり骨が折れる作業になりますが……），A4縦でもA4縦×2（見開き）でも解像度はお好みのままである．

また，高解像度だからといって，何もシステムフォントを24ドットにしなければならないということはない．フォントサイズが小さくても，画面がより広く使えればいいという考え方もある．セットアップ時にフォントサイズを選択できるようにして，ユーザーの希望に沿った使い方ができるようになれば，よりGoodだ．

こう書いているうちに，その昔PC-100というおもしろいマシンがあったことを思い出した．PC-100はディスプレイを横でも縦でも使うことができ，当時としては（いまでも!?）なかなかおもしろいパソコンではあったが先進的すぎたのであろう，あまり受け入れられなかった．いまなら同じコンセプトでWindowsやOS/2に対応させ，GUIマシンに仕上げることができればなかなか興味深いパソコンになるのではないだろうか……惜しいパソコンであった．

4 とにも，かくにも……

どうもこれまで日本のパソコン業界は，メーカー側の都合のよいほうへと流されてきたように思えてしかたがない．そろそろ本筋であるユーザー側へと，ソフト・ハードともに流れを変えて行かなければならないころだと思う．それがためにも「不満」「希望」など思うところがあればメーカーへリクエストし（方法はいろいろあるだろうが），固定概念にとらわれず，よりよいパソコン環境を決してメーカーためではなく自分のために創っていく必要がある．そのひとつとして，私は大きな声でいいたい「A4縦ディスプレイがつながる日本語環境がパソコンにも欲しい!!」と……．



NEXT STAGE

マイクロソフト（株）は，日本語Microsoft C6.0（以下MS-C6.0）の発売を記念してMicrosoft Language Conferenceを開催した．MS C6.0の概要などが説明されたほか，米国マイクロソフト社の言語事業部ゼネラルマネージャであるフレッド・グレイ氏より「Microsoftの目指す開発環境の未来」について講演があった．ここでは，それをベースにしながらマイクロソフトの言語戦略の一端を見てみよう．

マイクロソフトの言語戦略 開発環境のシステム化へ向かう

MS-C6.0は，PWB（Programmer's Work-Bench）を中心にした「プロジェクト指向」の統合開発環境であることが強調されている．エディタ，コンパイラ，リンカなどツールの集合体である開発環境が，PWBのような中心をもつことによってシステム化されていく流れは完全に定着したといえる．グレイ氏の講演では，システム化の流れは今後さらに大きなものになりそうである．

講演では，まず将来の方向性にかかわる4つのテーマがあげられた．

- GUI開発ツール
- オブジェクト戦略
- 開発環境
- 最適化技術

GUI開発環境とはおもにWindowsを指す．ここでは，現在の実質的な標準Cコンパイラであるms-CとWindowsSDKが今後も中心的な役割を果たしていくようだ．それを補完するものとして，ユーザーインターフェース作成ツールやより進んだデバッキング環境，さらにクラスライブラリの充実なども推し進められる．また，FORTRANやCOBOLによるGUI開発サポートが表明されたことは注目される．とくに，日本ではMS-CとWindowsSDKという限られた開発環境しかないだけに，Windowsをサポートする言語系が増えることは歓迎すべきだろう．

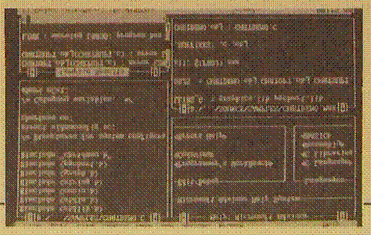
オブジェクト指向に関して，マイクロソフトはまだC++への対応を表明しただけの段階である．「C++は新しい言語である」というのがマイクロソフト社の考え方である．そのため，現在のC++の状況，CPUやOSの動向の検討な，どさまざまな要素が考慮されているところだという．MS-CなどがいずれC++化していくことは予想されるものの，クラスライブラリなど開発環境の整備にはまだ時間がかかりそうだ．コードビューのオブジェクト化という方向性も出てこよう．

開発環境では，それ自体のGUI化が進められるとともに，「プロジェクトデータベース」という概念が打ち出されている．MS-C6.0の「プロジェクト指向」をさらに推し進めるための中心になる概念であろう．コンパイラなどの開発ツール，コーディング，テストやデバッキング，プロジェクト管理などの中心に開発データベースを据えたシステム環境が将来像として検討されている．APIのデータベース上での公開なども考えられている．

今後マイクロソフトの言語プロダクトは，次のような方向をたどるとされる．

- すべての言語の統合環境化
- GUI開発ツールの充実
- 広範なオブジェクト戦略
- 最適化技法の徹底

日本における有力なCコンパイラは，MS-CとTurboCに絞られた観があるが，昨年MS-C6.0とTurboC++が登場した中で，両者の戦略的相違は明らかになってきた．マイクロソフトが統合環境の充実と最適化技術の推進におもきをおいたのに比べて，ボーランドはC++を取り込むという対応の早さをアピールしたのであった．マイクロソフトの「C++という新しい言語への総合的な取り組み」というスタンスには，業界標準コンパイラを擁する立場としての余裕を見ることができる．



ワンラインNEWS
NJKは効率的にソフトを開発するために，来年度から関東地区8か所のソフト開発センターにLANを設置する．これにより分散開発体制を整え，生産性を向上させる．

Illustration : Mikio Sasaki

APPLE II

アップルII

アップルIIは，1977年4月に売り出された．当時は，アルテア8800とその互換機であるIMSAI8080などが市場を風靡していた．そこに有名なスティーブン・ジョブズ氏が登場する．ジョブズ氏は，当時のパソコンはパソコン自体にさわるのが趣味の人が使っていると考えていた．だから，知識が豊富でプログラムなどの手間暇かかる作業も厭わない．むしろ，手間のかかる作業自身が，パソコンを買う目的であった．

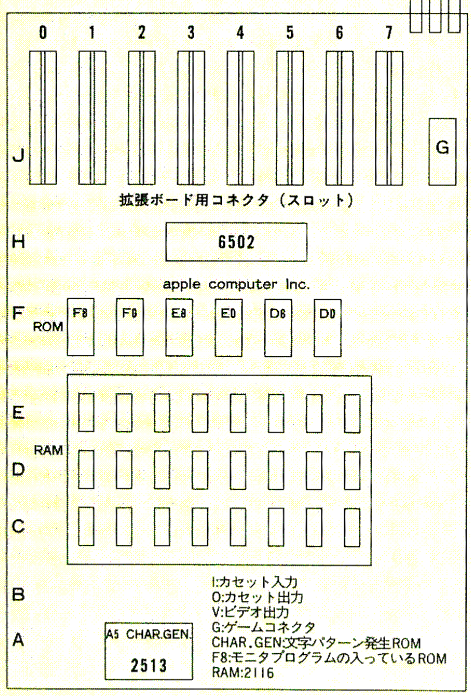
趣味で使うパソコンは，それほど大きな市場にはならず，多少不便であっても価格の安いものが必要だ．ジョブズ氏はそう考えたのだった．

たとえば，データの入出力装置は，大型コンピュータの端末とかASR33のようなテレタイプを使うしかなかった．当初アルテアは，データの保存に紙テープを使っていて，後にカセットテープを使うようになった．いっぽう，汎用コンピュータでは記憶装置として，磁気テープ，フロッピーディスクを使っていたが，高価でパソコンには向かなかった．このころ，テキサス州ダラス市のSWTP社が家庭用のテレビにタイプライタの文字を映し出す装置を作り，テレビジョンタイプライタという名で売りだした．

プログラムは，ほとんど機械語で作り，電源を落とすと消えてしまうので，また入力するというような具合であった．マイクロソフト社のビル・ゲイツ氏は，汎用機用に作られていたベーシック言語をアルテア用にポーティングしていた．しかし，これは500ドルもして，コンピュータ本体の値段と同じくらいであった．

このような市場で，ジョブズ氏がはじめて作ったアップルＩは，「アルテアのすべての機能を１枚のボードにのせた」といったものであり，「モノクロディスプレイしか使えず」また「マイクロプロセッサ6502が当時の標準と異なっていたためソフトウェアがまったく使えなかった」ことが欠点であった．他社と異なるものを選ぶことにより，自社の存在意義を世に問う，というのがジョブズ氏の方針であった．6502というマイクロプロセッサについては，次回に述べる予定である．

Apple II マザーボード概要



上部のコネクタのうち［I］はカセットレコーダからの入力［0］は出力．［V］はビデオターミナルへの出力．［J］列［G］の所はゲーム用のコネクタ．［6502］がメインのCPU．［F］列の［F8］が2Kのモニタ用ROM．［F0］［E8］［E0］が6Kベーシック．［C］［D］［E］列は48Kまで入るRAM．［A5］はビデオターミナルに行く文字用のキャラクタジェネレータ．

拡張スロットの使い方の例（50pin）

SLOT0	16K RAM・言語カード	SLOT1	（プリンタインタフェース）
SLOT2	（MODEM）	SLOT3	（80-カラム）
SLOT4	（CP/M）	SLOT5	（DISK3，4）
SLOT6	DISK1，2	SLOT7	（DISK5，6）

（）内のカードはどこに入れてもよい．

アップルIIは，以上のような経験の上に設計された．すなわち，

- パソコンにはキーボードがついていなければならない
- テレビがディスプレイとして使えなければならない
- テレビはゲームができるようにカラー化しなければならない
- 増設ボードは安く簡単に作られなければならない
- 言語はいちいちロードしなくてもプログラムを書けるようにする（ベーシックをあらかじめROMに書き込んでおき簡単にロードできるとよい――カセットテープは信頼性に欠けるうえ遅い）
- 軽くて熱が出ずファンの音もうるさくない電源（スイッチング電源の採用）

このような機械なら，技術的な知識のない人でも使える，市場性の大きいコンピュータができるとジョブ

ズ氏は期待した。

発表の舞台は，1977年4月の第1回ウェストコーストコンピュータフェアであった。このときの熱気については，『トランジスタ技術』の1977年7月号に寺田冬彦氏が書いておられる。アップルIIは，この年のうちに4000台を売り，アルテア，IMSAIに次ぐ位置にまで人気が上昇した。

上記の仕様をすべて満足させると，当時のコンピュータとしては最高級のいわばキャデラックのようなものになった。フロッピーディスクが使えることと，はじめからベーシックがついていたことで，アマチュアでもプログラム開発が行えるようになり，アプリケーションソフトの拡充が図れた。

バージョンアップの点でも，適切な手段を講じている。フロッピーディスクについては，ケンタッキーフライドコンピュータという会社が，はじめてパソコン用に使った。これを安価で便利なものに改良した記憶装置として，翌年1月のラスベガスでの消費者エレクトロニクスショーに向けて開発した。当初のベーシックは，整数のみであったので，割算をするとまずいことになった。そこで，実数を扱えるマイクロソフト社の「アップルソフトベーシック」に積み替え，アップルIIプラスとした。

いっぽう，販売面でもアップル社はディーラーを育てていった。たとえば，資金力の不足している小売業者に共同広告の手段で広告費の一部を負担した。

このように大きな，また当時としては先の見えないプロジェクトを引っ張っていった偉大なリーダーがスティーブン・ジョブズ氏であり，技術的にその夢をひとつひとつ実現していったのが，スティーブン・ウォズニアック氏である。

アップルIIの成功の秘訣は，「その時代に手にはいる技術プラスアルファにより，購入可能な金額の範囲内で最高の組み合わせのものを作りだした」ということにありそうである。

Illustration : TOSHIHITO ITO

特集

Windowsの時代である!

WinDowS PROGRAMMING

Windowsプログラミングを徹底解説・大解剖

1991年は，日本においてWindowsが本格的に普及する年になりそうだ．年頭から春にかけて，各メーカーから日本語Windows 3.0が続々と出荷されことになるだろう．

昨年5月に米国で発売されたWindows 3.0は，昨年中に約300万セットを販売したといわれる．米国とはかなり状況が異なるため，日本においてもおなじように爆発的な普及がみられるかは疑問ではあるが，Windows 3.0の製品の質やハードウェアの状況，ソフトウェア市場にほかの大きなニュースがないことなどを考えると，業界で台風の目になることはまちがいないだろう．

しかし，Windowsが普及しても第一に問題になるのが，アプリケーションの不足である．Windows 3.0に対応したアプリケーションソフトは，米国では昨年中に1500タイトルをこえたとされるが，日本では大幅に少ない数字になりそうだ．日本で製作したWindowsアプリケーションはもとより，米国などから移植されたアプリケーションもそれほど数はない．Windowsの普及に対応してアプリケーションを充実させることが急務になると思われる．

そのなかで，Windowsプログラムを扱えるプログラマが少ないという現在の状況は大きな課題となりそうだ．独自のアプリケーションを開発するにしても，また米国のWindowsアプリケーションを移植するにしても，Windowsアプリケーションの開発環境を理解しているプログラマが求められる．

DOS上のCプログラムの常識が通用しないWindowsプログラミングとはなんだろう．Windowsをめざすプログラマたちのために，Windowsプログラミングを徹底解説・大解剖してみよう．

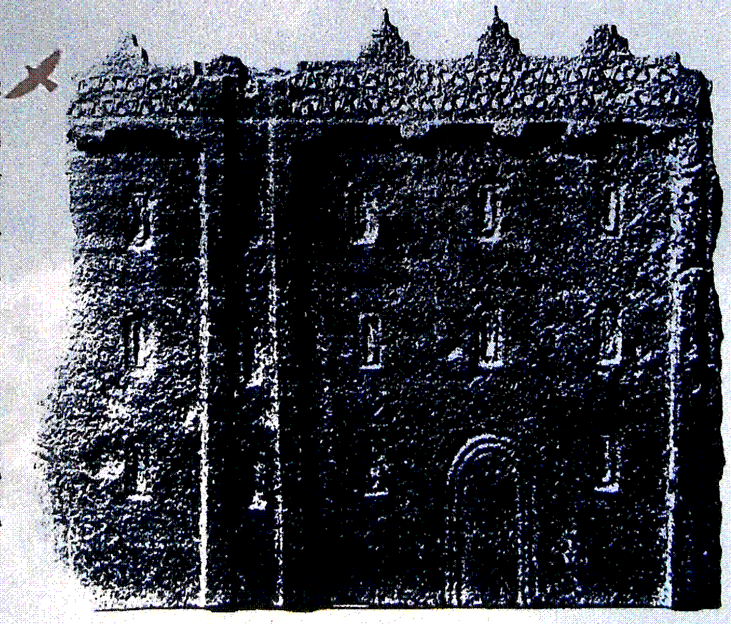
窓の歴史

History of Windows

建築の窓

古代の窓

「窓」とは，建物の開口部で出入口ではない部分の総称である．その起源は非常に古く，建物の起源とほぼ同じだと推測されている．エジプト，アッシリアの遺跡や壁画にも「窓」を見つけることができる．時代はずっとくだるが日本にも，古代の窓が存在する．現存する世界最古の木造建築物である奈良法隆寺の回廊にみられる連子窓である．それより古いものとして，1982年に発掘された飛鳥山田寺の遺構があり，その回廊跡から法隆寺のものによく似た連子窓が発見されている．時代的には，法隆寺より1世紀ほど遡った7世紀中期の建物である．



窓と建築構造

西洋の石造りを基本とした建築物と日本の木造建築とでは，窓に対する構造上の考え方は大きく異なってくる．石造建築物においては，みずからの重さをささえるための構造が要求され，必然的に窓が小さいほうが構造上は堅牢であることになる．それに対して，日本のような木造建築物では，「窓をあけるのではなく，もともと空いているところをかりにふさぐ」(若山滋『「組み立てる文化」の国』1984，文芸春秋）という考え方で窓が設けられた．それが，間戸や間処と書かれる「まど」の語源になっているとされる．

和風建築における窓の起源をもっと遡れば，おそらく蔀戸（しとみど）にいきつくであろう．蔀戸とは，木製のあけることのできる壁なのである．

「光る壁」の意味

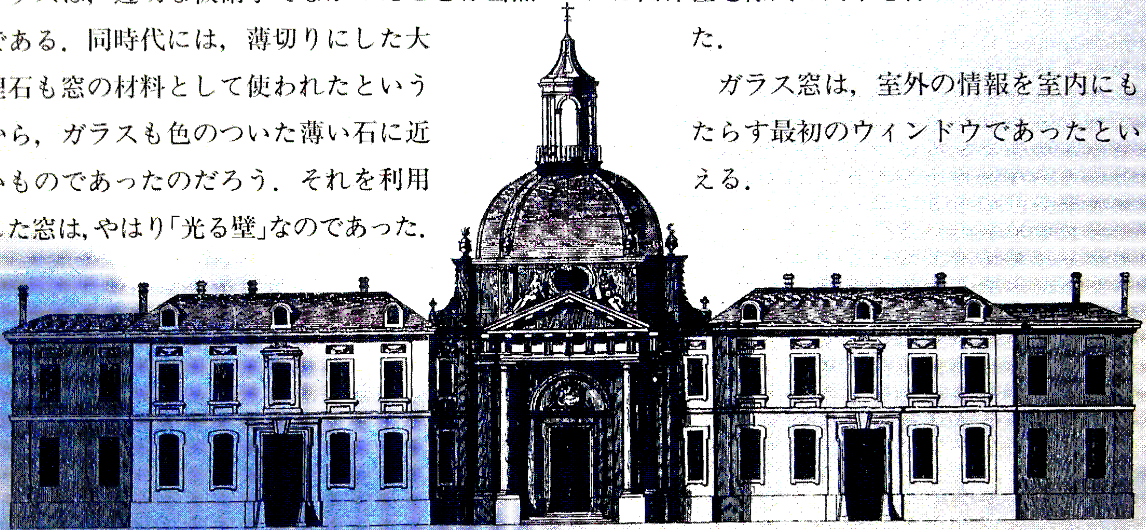
和風建築において明障子が出現したとき，それは現代的な窓という概念とは異なり，「光る壁」として認識された．構造上は対極的な関係にある西洋の建築物でも「光る窓」は存在した．ガラス窓の出現である．

ガラスは古代オリエントの時代からみられるが，窓に利用されるようになったのは，ローマ

帝政期からのことである．ベスビオス火山の噴火で埋もれた町として知られるポンペイからはブロンズの窓枠さえ出土している．この時代のガラスは，透明な板硝子でなかったことは当然である．同時代には，薄切りにした大理石も窓の材料として使われたというから，ガラスも色のついた薄い石に近いものであったのだろう．それを利用した窓は，やはり「光る壁」なのであった．

も借景という形で「見晴らし窓」は存在したが，わずかに権力者や富豪の趣味として可能であっただけであり，住宅建築に使われるようになるには西洋住宅様式の到来を待たねばならなかった．

ガラス窓は，室外の情報を室内にもたらす最初のウィンドウであったといえる．



ローマ帝政期において，窓は初期キリスト教会で宗教的な演出のために利用され発達した．それが「光る壁」と結びつき，大理石の枠に色ガラスをはめる方法がとられるようになったのはビザンティン帝国でのことだった．これが，ステンドグラスの原型である．

その後，ゴシック建築によって窓の面積は大きく増大することになる．ゴシック建築の合理的な構造は石造建築物を発展させ，窓を大きくとることを可能にしたのだ．さらに，それが「光る窓」の宗教的効果と結びついて，中世キリスト教会のステンドグラス全盛時代へとつながるのである．

外景をもたらした「見晴らし窓」

ルネサンス以降，窓は建築様式の変化とともに多様な形をとるが，近代工業の登場によって板ガラスの量産が可能になると窓も大きな影響を受けた．ピクチャウィンドウ（見晴らし窓）が住宅に使えるようになったのである．日本に

異質な窓の時代

近代工業が，ガラス窓の次に室内に持ち込んだ「窓」はテレビジョンであった．テレビジョンは，建築の窓から考えると異質な存在ではあるが，外の情報をもたらす「窓」という意味で，ピクチャウィンドウの延線上にとらえることができる．これは，隣あった景色を見るだけでなく，まったく離れたところから情報を運んでくれる「窓」であった．テレビジョンという「窓」は，登場するとまもなく家庭住宅の中でもっとも注目される「窓」となった．

しかし，テレビジョンの情報は，そこに映し出される情報を見るという意味で，まったく受動的なものである．人はそれに満足しないで，対話のできる「窓」を作りだした．対話のできる「窓」は，住人が好きなことを書ける紙になったり，ほかの人との通信をしたり，ゲームをしたりできる．新しい「窓」は，コンピュータと呼ばれる．

コンピュータの窓

「情報の窓」の発展

テレビジョンやコンピュータのディスプレイは，「情報の窓」として位置づけられる．つまり，ディスプレイはもともとフルスクリーンのウィンドウなのである．コンピュータ技術が発展する中で，ディスプレイという「窓」をさらに多くの「窓」に分割して，さまざまな世界をそれにおさめようとする動きがでてきた．

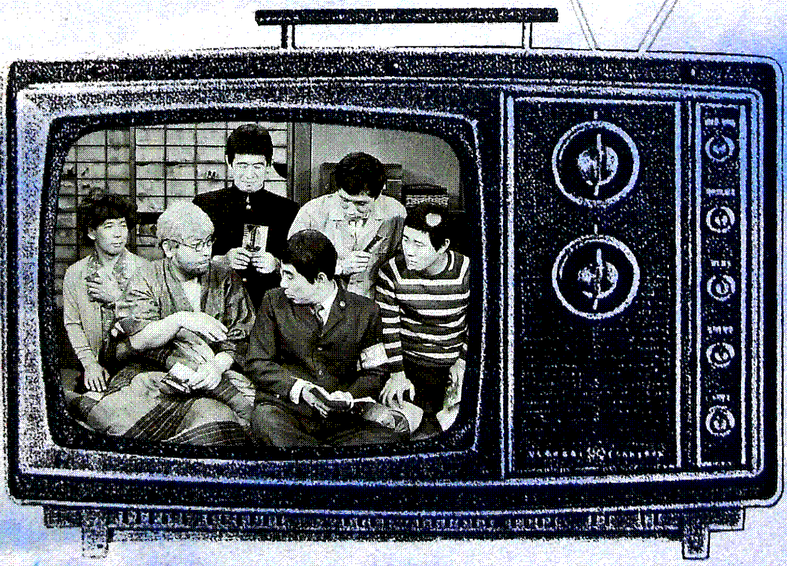
最初は，矩形の画面を上下，左右に2分割または4分割して，異なる情報を表示させるという手段がとられた．これらは，その機能を持つエディタソフトや表計算ソフトなどのプログラム固有の機能であった．もちろん，共通した決まりを作るほど複雑な「窓」でもなかった．

「窓」システムの誕生

初期のコンピュータの画面は，「窓」の構造としては単純なものであったが，「単純な窓」はそれを使うものにとって，見晴らしの悪い「難解な窓」でもあった．コンピュータの「窓」が発達したのは，見晴らしのよい窓をつくろうという努力によるものである．

西洋において「建築工事の歴史は，強度をおとさずに大きな窓をつくるための努力の歴史であった」（『カテドラルを建てた人々』ジャン・ジャンペル著，飯田喜四郎訳，1969，鹿島出版会）という言葉があることになぞらえるなら，「コンピュータウィンドウの歴史は，実行速度を落とさずにより高密度なウィンドウをつくるための努力の歴史であった」というべきだろうか．

その努力は，米国カリフォルニア州にあるゼロックス社のパロアルト研究所（PARC）ではじまった．1973年，PARCで最初の個人用コンピュータといわれるアルトが誕生した．アルトは，ウィンドウシステムをもった総合的な開発環境であるSmallTalkを搭載し，マッキントッシュやMS-Windowsなどのさきがけとなったのである．SmallTalkを起源とするウィンドウシステムは，よく似た外見とアーキテクチャをもつことになる．



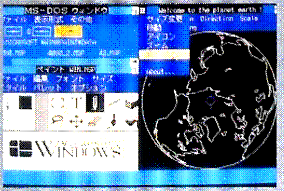
©日本テレビ，「シャボン玉ホリデー」

Windows登場

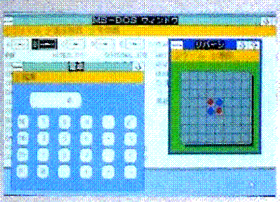
ウィンドウシステムを持つパソコンとして，最初に市場に大きな影響を与えたのは1983年に登場したアップル社のリサであった．1984年には，マッキントッシュ128Kが登場し，やがてマッキントッシュシリーズとして一大勢力を成すようになる．その時点ではまだ市場の多くを占めるDOSマシンはウィンドウシステムから遠い存在であった．その巨大なマーケットにウィンドウシステムを持ち込んだのが，ほかならぬMS-Windowsだ．

MS-Windowsの歴史

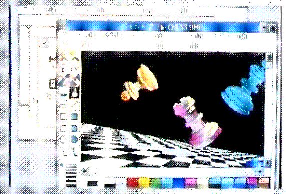
MS-Windowsの開発とリリースがアナウンスされたのは1983年のことだった．アップル社のリサもこの年に登場しており，パソコンのグラフィックユーザーインターフェース元年ともいうべき状況にあった．実際にPC/ATとその互換機向けにWindows 1.0が発売されたのは1985年のことだった．翌年には，日本でもPC-9801用のWindows 1.0が発売されている．



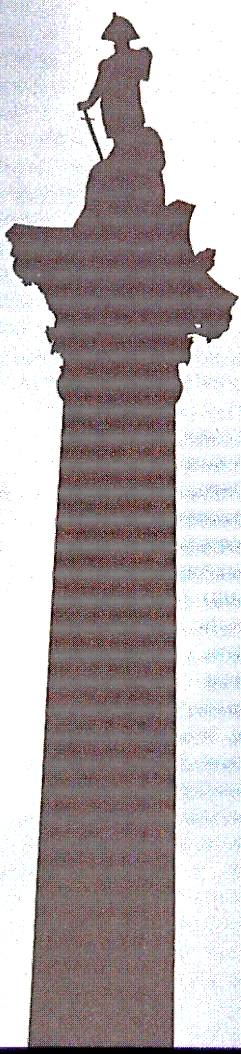
Windows 1.0



Windows 2.1



Windows 3.0



Windows 1.0

Windowsシステム自体は基本的に現在のWindowsと同様である．メモ帳，カレンダー，時計などのデスクトップツール類がバンドルされたのも1.0のときからだ．大きな違いは，1.0ではタイルドウィンドウが採用されていたことだ．システムリクエストは次のようなものであった．

- MS-DOS 2.0（以降）
- メモリ：384KB以上（複数のアプリケーションを動作させるためには512KB以上）
- フロッピーディスクドライブ2ドライブ（またはハードディスク）

PC-9800シリーズでいえばVMなどの全盛期であり，現在のように最低640KBは搭載している状況ではなかった．国内では，80286マシンがほとんどなく，ハードディスクはまだ高価だった．米国でも80286が次第に普及している程度だった．

そうしたハードウェア環境でMS-Windowsを動作させるのはかなりきびしく，実行速度，メモリの制約など実用化にあたって問題は多かった．

Windows 2.0

Windows 2.0が発売されたのは1987年のこと．新しいバスアーキテクチャを持ったIBMのPS/2，プロテクトモードメモリを利用するOS/2が発表された年であった．日本語版は，88年に登場している．Windows 2.0は，オーバーラップウィンドウが採用され，ユーザーインターフェースは大きく改善された．実行速度の改善とEMSによる拡張メモリのサポートも行われた．

当時は，次世代OSとしてOS/2への期待が大きく，MS-DOSからOS/2への移行がパソコン界にドラスティックな変化をもたらすと考えられていた．MS-DOSのGUIとしてMS-Windows，OS/2のGUIとしてプレゼンテーションマネージャ（PM）という構図ができていた．また，CPUアーキテクチャの利用とWindows 3.0への予兆を感じさせるものとして，Windows/386の存在を忘れるわけにはいかない．Windows/386は，Windowsが単なるMS-DOSのGUIにとどまらないことを示していたのである．

Windows 3.0

1990年5月に米国でリリースされたWindows 3.0では，メモリ環境の大幅な改善がみられた．リアル，スタンダード，エンハンストの3モードを用意し，プロテクトメモリや386の仮想マシン機能の利用を可能にし，CPUの能力を引き出すことに成功した．さらに，ユーザーシェルや画面デザインを変更することでGUIの特性をより強く打ち出している．ネットワーク対応が強化されたことも見逃せない．この春までには各社から日本語版が出そろいそうだ．

Windows 3.0によって，MS-DOS，WindowsとOS/2 PMとの関係は微妙なものになってきた．MS-DOSに慣れた多くのユーザーは，OS/2という新しいOSに取り組まなくても，CPUの資源を利用することができるようになったのである．MS-DOSからOS/2への移行というテーマは，MS-DOS，WindowsとOS/2 PMの住みわけというように変更されたようである．

特集・Windowsの時代である！

Windowsプログラミングを徹底解説・大解剖

MS-Windowsのプログラムは，いままでMS-DOS上で動作するプログラムをつくってきたプログラマにとって異質なものだ．Windowsシステムの概念を知ったうえで，Windowsプログラムの作法を学ぶ必要がある．Windowsプログラムのむずかしさとは，一般のMS-DOSプログラムとの作法の違いによるところが大きい．まったく異なる種類のキーボードを急に使わなくてはならなくなったようなものである．いままで，慣れ親しんだ環境がむしろ障害になってしまう．

しかし，Windowsのプログラムは，それ自体が困難なわけではない．今までの常識が通じない新しい環境であるがゆえの困難さである．だから，Windowsシステムと Windowsプログラミングを知ることで，すべてが解決されるのである．

Windowsの時代はもうはじまっている．概念からプログラムの構造まで，Windowsプログラミングを徹底解説・大解剖したこの特集で，Windowsの世界へ接近しよう．

Windows 3.0 登場

Windows 3.0がパソコン環境に新しい展開をもたらす

Windows 3.0 が昨年米国で発売されて以来，爆発的なブームを巻き起こしたというニュースはまだ記憶に新しい．Windowsシステムの普及とアプリケーションの増加は良循環を引き起こし，PCマーケットに確固とした橋頭堡を築いたようだ．これを「臨界点に達した」と表現する関係者もいる．米国でWindows 3.0が歓迎された理由には次の3つが考えられる．ひとつは，画面のデザインやシェルの操作性が一新されたこと．もうひとつは640KBというDOSのユーザーメモリの制限をうち破る新たなメモリ管理機構を採用したこと，そして豊富なWindowsアプリケーションが提供される見通しがついたことだ．

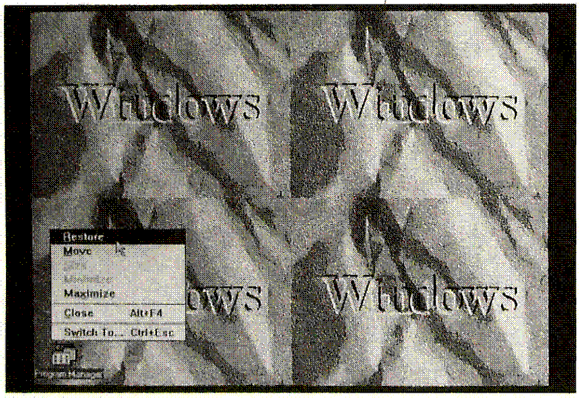
日本においてもWindows 2.xまでは，期待の大きさに反して実際の普及はおぼつかなかったのだが，Windows 3.0はこれまでの壁を打ち破ってくれるのではないかという期待がたかまっている．ここではWindows 3.0のグラフィカルなユーザーインターフェースと，画期的なメモリ管理機構，日本語環境での見通しなどを紹介する．

デザインと機能を一新して よりユーザーフレンドリーな 環境へ

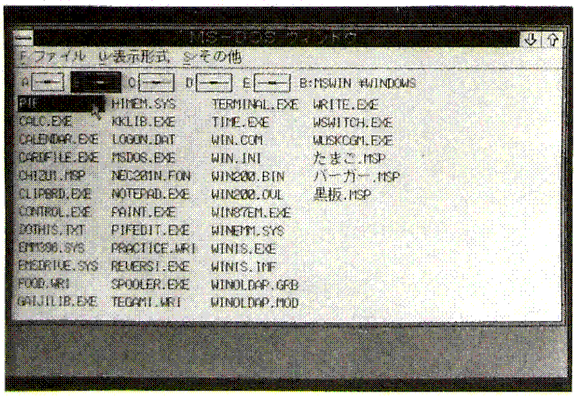
まずはWindows 3.0の画面を見てみよう．Windows 2.11の画面と比較すると大きく異なることは一目でわかるだろう．立体的なボタンやカラフルなアイコン，アルファベットについてはプロポーショナルフォント（注１）になり非常に読みやすい．

Windows 3.0では，「プログラムマネージャ」が標準シェルになっている．Windows2.Xのシェルである「MS-DOSウィンドウ」はMS-DOSのファイル一覧をそのまま窓枠に押し込めたようなものだった．DOSのファイル名に関する知識がないとWindows 2.Xは使えなかったのである．グラフィックと呼ぶにはあまりに中途半端な「MS-DOSウィンドウ」に比べると，デザイン的にもずいぶんと洗練された．ようやくGUIの名にふさわしい外観を備えたという感じだ．

（注１）プロポーショナルフォント　隣あった文字の間隔を調整して文字列全体が整って見えるようにした書体のこと



画面１　バックグランドに好みのグラフィックデータをロードできるようになった．これをウォールペーパー（壁紙）という

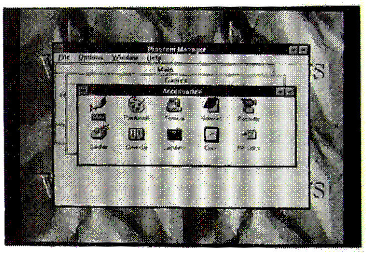


画面２　Windows 2.xは起動すると「MS-DOSウィンドウ」が現れる

プログラムマネージャ

Windowsを最初に起動すると「プログラムマネージャ」が表示される．プログラムマネージャは，Windows 3.0の標準シェルであり，Windowsシステムの中から任意のプログラムを実行するスターターの役割を果たす．プログラムマネージャではプログラムをアイコンベースで取り扱う．目的のアイコンにマウスを移動してダブルクリックすれば，そのプログラムを起動することができるので視覚的でわかりやすい．

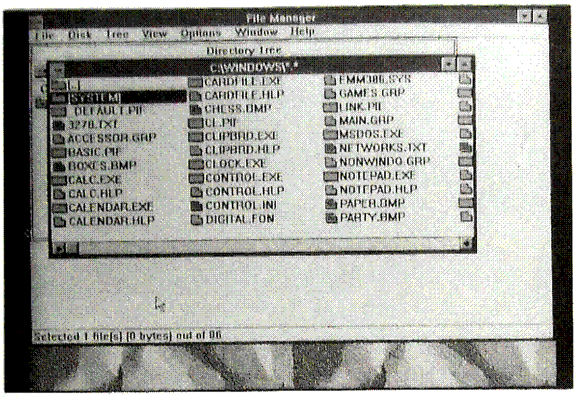
また，プログラムマネージャにはプログラムを用途や機能に応じてグループ分けする機能がある．最初の設定ではあらかじめ「メイン」「アクセサリ」「ゲーム」という３つのグループが存在する．「メイン」グループには「ファイルマネージャ」「コントロールパネル」をはじめとするWindowsのシステム関連のプログラムが，「アクセサリ」グループには「ライト」（簡単なテキスト編集ツール）や「ペイント」（描画ツール）「ターミナル」（通信ソフト）などのデスクアクセサリ類，「ゲーム」グループには「リバーシ」「ソリテア」などのゲームプログラムが含まれる．


画面３　メイン，アクセサリー，ゲームプログラムグループをカスケードで表示

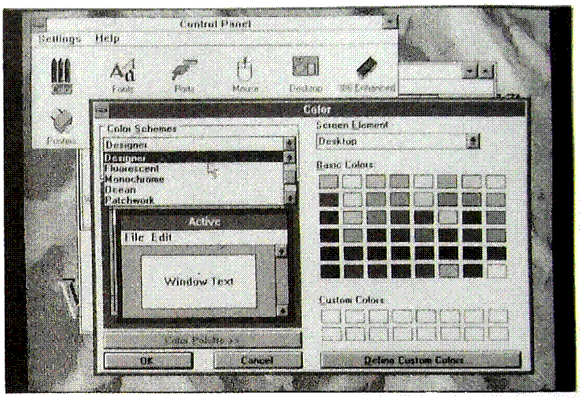
各グループはプログラムマネージャの子ウィンドウとして，１グループが１ウィンドウで表示される．ウィンドウの大きさや配置は自由に変えられるし，グループ自体をアイコンにすることもできる．

利用者は現在のグループにプログラムを追加したり，新しくグループを作ったりしてプログラムマネージャをカスタマイズできる．グループという概念は，そのプログラムの含まれるドライブやディレクトリとは独立している．つまり，ひとつのプログラムを複数のグループに所属させることもできるし，「自分がよく利用するものだけを集めたグループ」や「進行中のプロジェクトに関わるものだけを集めたグループ」などを自由に設定できる．これによって，ユーザーはMS-DOSのディレクトリやパスの知識がなくてもプログラムを実行できる．

Windowsプログラミング用語
Windows関数（Windows function）　Windows内にあらかじめ用意されていて，ユーザープログラムから呼び出させるWindowsの内部ルーチン．



画面４　ファイルマネージャ



画面５　コントロールパネル．画面色設定のウィンドウが開いている

プログラムマネージャでプログラムとして登録できるのは次のようなものである．

- MS-DOSの実行形式のファイル（.EXE, .COMなど）
- MS-DOSのバッチファイル（.BAT）
- PIFエディタ（注２）で定義されて，起動可能となっているファイル
- WIN.INIであらかじめ拡張子が登録されていて，拡張子によってどのプログラムが起動するか決まるようなデータファイル

（注２）PIFエディタ　MS-DOSアプリケーションをWindows上で動作させるための情報を記述するファイル（PIFファイル）を作成するツール

ファイルマネージャ

マウスポインタをプログラムマネージャのメイングループに含まれるファイルマネージャのアイコンに移動してみよう．ここでダブルクリックするとファイルマネージャのウィンドウが現れる．ファイルマネージャはその名の通り，ファイルを管理するためのツールで，ディレクトリ中のファイルの一覧表示や，ファイルのコピー，移動，削除，ファイルやディレクトリの名称変更，ファイルの属性変更，ファイルの検索，プログラムの実行，テキストファイルの印刷などが行えるようになっている．

ファイルマネージャには，２種類の子ウィンドウがある．まず最初にハードディスク，フロッピーディスク，RAMディスクなどに存在するディレクトリがツリー形式で表示される．これが「ディレクトリツリーウィンドウ」だ．ウィンドウの上部には各ドライブがアイコンで表示されているが，そのドライブがハードディスクなのかフロッピーディスクか一目で区別できるようになっている．また，RAMドライブやネットワークドライブのアイコンも識別して表示する．

マウスをディレクトリ名に移動してダブルクリックすると，ディレクトリ中のファイルやサブディレクトリが表示される．これが「ディレクトリウィンドウ」である．ファイルの表示形式は名前だけを表示する，バイト数やタイムスタンプ，ファイル属性なども同時に表示することもできるし，表示する順序も名前順，拡張子順，サイズ順，タイムスタンプ順などに変更できる．ワイルドカードを使って表示するファイル名を限定することも可能だ．

また，画面でもわかるように，Windows 3.0からはディレクトリ名やファイル名にもアイコンがついた．アイコンは全部で４種類ある．

- ディレクトリアイコン：ディレクトリまたはサブディレクトリ
- 実行可能ファイル用アイコン：.EXE, .COM, .BATの拡張子のファイル
- ドキュメントアイコン：プログラムと関連づけられているデータファイル．データファイルを指定すると，特定のプログラムが起動する．
- その他のファイルを示すアイコン

実際のファイル操作は，ファイルマネージャの「ファイル」メニューを使って行うが，ファイルの移動などはマウスだけで簡単に行えるようになった．移動先のディレクトリウィンドウを画面に表示して，移動するファイルをクリックし，[CTRL]を押しながら移動先まで引っ張っていけばよい．

コントロールパネル

コントロールパネルは，画面の配色やハードウェア環境の変更などを設定する機能を持つ．プログラムマネージャのメイングループにあるコントロールパネルのアイコンをダブルクリックすると起動する．Windows 2.Xにも CONTROL.EXEというプログラムがあったがとても同じものとは思えないほどスタイルを一新した．設定項目

Windowsプログラミング用語
アイコン（Icons）　プログラムの種類や状態，処理，警告などをユーザーに視覚的に伝える小さなシンボルマーク．

ごとにアイコン表示されているので，そこをダブルクリックすれば設定ウィンドウが開かれる仕組みだ．アイコン表示されるのは，そのシステムで使っているものだけだ．たとえば，386マシンでなければ，386Enhancedのアイコンは表示されない．

Color

Windowsの各要素の色を設定する．従来よりも指定がずっとしやすくなった．まず色を選択し，サンプル画面に表示されたWindowsの各要素（アクティブタイトルバーやメニューなど）をクリックすればただちに変更できる．

さらに，ひと通り各要素の色を設定し終ったら，それをひとまとめにして「カラースキーム」として名前をつけて保存しておくことができる．Windowsにはあらかじめ「ラグビー」「オーシャン」「パステル」などのカラースキームが用意されていてその中から選ぶこともできる．

Desktop

画面の背景やカーソルのブリンクスピード，アイコン同士の間隔，アイコンやウィンドウの位置の基点となるグリッドの間隔設定，ウィンドウ枠の太さなどを設定する．ウォールペーパーの設定もここで行う．米国ではパソコン通信でウォールペーパー用のデータをアップロードするのが流行しているそうだ．

Printers

プリンタドライバの設定や選択，プリンタの用紙サイズや印刷方向，フォントの選択などプリンタに関わる設定を行う．

Ports

通信ポートの設定を行う．通信ポートを選択すると，そのポートでの通信条件の設定ウィンドウが表示されて，ボーレート，データビット長やパリティ，ハードフロー制御などの項目が設定できるようになっている．

Network

Windows 3.0ではLAN上での利用も十分考慮されている．ネットワーク用のドライバがインストールされているときは，ここでサーバーへのログイン，ログオフ，ユーザーID，パスワードなどが設定できる．

Fonts

画面表示用，プリンタ出力用のフォントを登録，削除する．同じフォントに複数の大きさの文字がある場合は，それも表示される．

International

国名，言語，キーボードの種類，長さの単位，時間や日付の表示方法などを設定する．

Mouse

マウスの移動速度，ダブルクリックの間隔，左右ボタンの切り替えの設定．

COLUMN

プロテクトメモリにアクセスする

80286以上のCPU（80286/80386/80386SX/80486）には実アドレスモード（リアルモード）と保護仮想アドレスモード（プロテクトモード）という2種類の動作モードがある．リアルモードは8086という以前のCPUとの互換性をとるためのモード，これに対してプロテクトモードは，マルチタスクシステムに求められるメモリ管理機能，保護機能，タスク機能などをハードウェア上で実現する機能を持っている．

MS-DOSは8086CPUマシンのOSとして誕生した．これまでのDOSアプリケーションはそのほとんどが80286以上のCPUであってもリアルモードで動作していると考えてよいだろう．しかし，セグメント：オフセット形式でアドレッシングして物理アドレスにアクセスする8086CPUの方式では1MBまでのメモリ空間しかアドレッシングできない．複雑化するアプリケーション，増大するデータという状況下で，メモリ不足を解消するためにEMSのようなメモリ拡張方式が提唱されてきたが，現実にはEMSメモリ対応ソフトのほとんどがデータ領域としてしか使っていないため，根本的なメモリの不足という問題は解決されなかった．

そこで考えられたのがプロテクトモードを利用するという手段である．米国では早くから，プロテクトメモリを利用するDOSアプリケーションプログラムを開発ためのツール（DOSエクステンダー）が使われている．Lotus1-2-3やdBASEIVといった有名なアプリケーションプログラムでもこのDOSエクステンダーが使われている．DOSエクステンダーを使って開発されたアプリケーションプログラムそのものの中にプロテクトメモリを利用するための仕組みが組み込まれてしまうので，エンドユーザーはEMSのようにメモリの利用方法を意識する必要はなくなる．

このようなプロテクトモードメモリ活用の流れは複数のアプリケーションがプロテクトモードでメモリを共通に利用できるような，インターフェースの策定へと動いていった．最初に提唱されたのはVCPIと呼ばれる方式である（1989年1月）．現在はVCPIではマルチタスク環境には不十分だとするマイクロソフト社など11社がDPMIという新たな規格を発表している（1990年5月に0.9版）．Windows 3.0もこのDPMIに準拠している．

Date/Time

システム日付, システム時刻の設定.

Keyboard

キーボードを押し続けたときの反復入力のスピード設定.

Sound

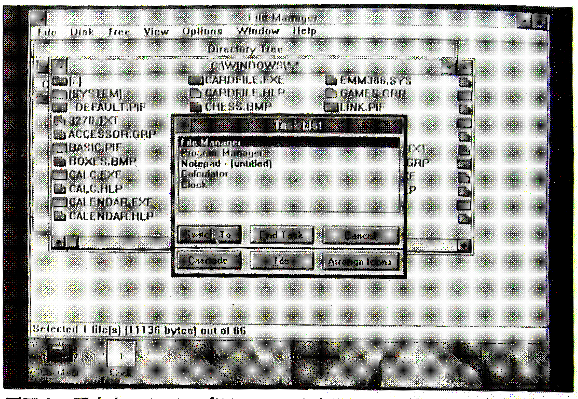
エラー時にビープ音を出すかどうかの設定.

386Enhanced

Windowsが後述する386エンハンストモードで起動したときのみ表示される. Windows用に作られていないアプリケーションを386の仮想86モードを使って実行するときに必要な設定を行う.

タスクリスト

Windows 3.0ではメモリ管理機構が改善されたことによって, いっそうマルチタスク環境としての意味あいを強めた. 今回のバージョンアップでは, アプリケーションを切り替えるためのツールとして新たにタスクリストが導入された.



画面6 現在走っているアプリケーションを表示し, 切り替えることができる

従来Windowsでは一番手前のウィンドウ（アクティブウィンドウ）のアプリケーションだけが直接操作でき, 別のアプリケーションに移るときは, そのウィンドウ（アイコン化されているときはアイコン）をクリックしてアクティブウィンドウにするという方法をとっていた（キーボードでは[ALT]+[ESC]によって画面を切り替えることもできるが). しかし, たくさんのウィンドウを開くと, 表示できる範囲の狭いパソコンの画面ではすぐに後ろに隠れて見えなくなってしまうものが出てくる. タスクリストは現在動作中のすべてのアプリケーション名

COLUMN

WindowsアプリケーションとDOSアプリケーションの違い

Windows 3.0では, MS-DOS上のプログラムを実行することができる. とくに386エンハンストモードを使えば, 複数のMS-DOSマシンを操作するような環境を手に入れることだってできるのに, Windows専用のアプリケーションが必要なのかという疑問を持つ読者もあるだろう. 利用者側から見た場合, Windowsアプリケーションには次のような特徴がある.

①動的なデータ交換が可能

Windowsには従来からクリップボードと呼ばれる機能があった. あるWindowsアプリケーションでデータのコピー(またはカット）を実行すると, 選択した範囲は「クリップボード」に複写される. ペーストを実行すると, クリップボードの内容が貼り込まれるのである. ペースト先は別のアプリケーションであってもかまわない.

動的なデータ交換（DDE：Dynamic Data Exchange）とは, これを一歩進めたものでコピー元のデータの変更がペーストした先のアプリケーションにまで伝えられる. たとえば, Excelのグラフを Wordにコピーしたとしよう. Excelでグラフの元となるデータを変更した場合, Wordに張り付けたグラフも変化するのである.

②操作方法・環境が共通

Windowsアプリケーションのユーザーインターフェースはすべて Windows本体と共通である. ウィンドウの開き方, 閉じ方, メニューの選び方, 複数の項目から選択するやり方などはどのアプリケーションでも同じである.

DOSアプリケーションでは, それぞれのアプリケーションプログラムごとに「環境設定」を行う必要がある. 使用しているプリンタの機種やマウスの種類を登録する作業はWindowsアプリケーションでは必要ない. 周辺装置との接続状況などはWindowsが一括して管理しているからだ.

③Windowsの動くパソコンならアプリケーションは同じでよい

機種ごとのハードウェア的な違いはWindowsが吸収する. 複数のメーカーのパソコンを利用するユーザーは各機種専用のWindows 3.0が必要だが, アプリケーションは共通に使える.

Windowsプログラミング用語

アクセラレータキー（Accelerator keys） 短縮キーのことで, 任意のキーストロークによってマウスのクリックなどと同じようにコマンドメッセージを生成する.

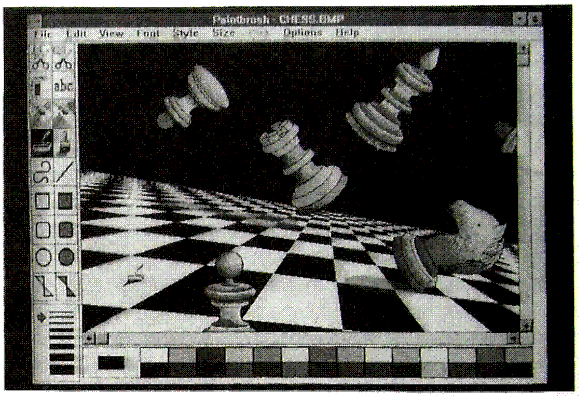
を表示して，どれをアクティブにするかを選択できるようにしたものだ．

タスクリストを表示するには，①［CTRL］キーを押しながら［ESC］キーを押す，②コントロールメニュー(画面左上の細長い四角をクリックすると表示される）で「Switch to」コマンドを選択する，という２つの方法がある．タスクリストでは現在実行中のアプリケーションの一覧が表示される．この中から選択したいアプリケーションを選べばよい．またタスクリストでは，アイコンの並べ替えやプログラムの終了，ウィンドウ表示画面を重ね合わせ（Cascade）から整列（Tile）に切り替えることもできる．

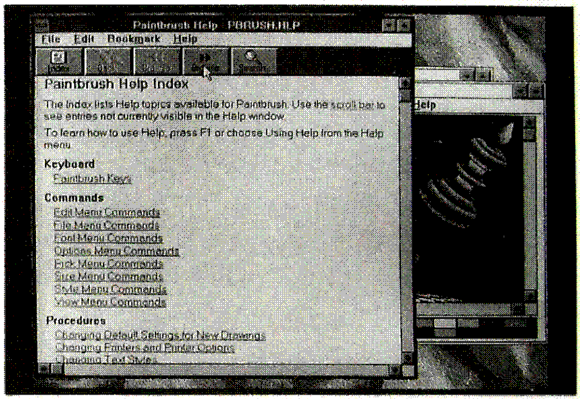
そのほかのアクセサリ・ツール

そのほかのユーティリティもそれぞれ機能強化されている．ノートパッドで取り扱えるファイル容量が50KBまでに拡大された．付属の通信ソフトにはKermit，Xmodemなどのプロトコルがサポートされることとなった．描画ツールとしては従来のMicrosoft Paintの代わりに機能の高いPaintbrushを採用することによって，カラーのイメージデータも取り扱えるようになった．電卓には８進数，16進数のモードが加わり，論理演算も可能となった．時計はアナログ表示とデジタル表示の２通り．そのほかに注目すべき新ツールとしてRecorderがある．これは一種のマクロレコーダーで，一連の連続したキー操作やマウスの移動をファイルに記憶するものである．

Windows 3.0では，ヘルプ機能も大幅に改良された．ヘルプはプログラムマネージャやファイルマネージャからWINHELP.EXEを実行して呼び出すこともできるし，それぞれのアプリケーションでHelpメニューを選択してもよい．呼びだした画面に応じたヘルプメッセージが表示されるだけでなく，インデックスを表示させて他の画面のヘルプを見ることも可能だ．特定の語句（点線でアンダーラインが引いてある単語）についてはクリックすればその定義を参照することができ，特定の言葉（実線でアンダーラインが引いてある単語の並び）についてはさらに詳細な説明を参照できるといったハイパーテキスト風の機能も備えている．WINHELPの恩恵を受けるのは利用者だけではない．アプリケーション開発者もWINHELPのフォーマットに従って作成すれば，構造化されたヘルプファイルを提供できることになる．



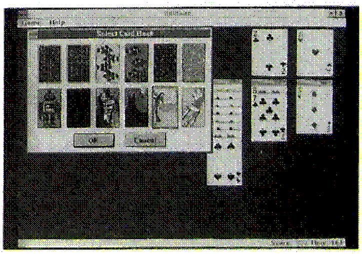
画面７　カラービットマップを扱えるPaintbrush



画面８　Windowsアプリケーションは共通のヘルププログラムを利用できる

一歩人間に歩み寄った環境に

マイクロソフトはWindows 3.0によってパソコン上のGUIについて，ひとつの回答を示したといえるだろう．メニュー構成ひとつとっても，非常によく考えた上で変更されているという印象を受ける．コントロールパネルのように，これまで３つのメニューさらにそれぞれがサブメニューを持つという複雑な形式をやめて設定項目ごとにアイコン化するという方式に変更したもの，Paintbrushのようにこれまでのプログラムを捨てて新規ソフトを導入したものもある．ウォールペーパーやカードゲームに見られるように，「遊び心」をくすぐる要素も増えている．どの変更点をとってもおおむねセンスがよくなった，親しみやすくなったということは確かだ．



画面９　Solitaireはカードの模様を変えられるなど工夫を凝らしたゲームだ

CPU性能をフルに活かす3つのモード 386用と286用を一本化

Windows 3.0登場の意義はその外観だけにあるのではない。もうひとつ見逃してはいけないのが，メモリ管理方式の変更である。Windows 3.0はプロテクトメモリを利用する仕組みを備え，OSレベルで640KBというメモリの壁を超えることになったのだ。

Windowsは8086系のマシンで動作する（とはいえ，2.X以前のバージョンでWindowsを使おうとすると，CPUの速度が速いものでなければ使いモノにはならなかったのだが）。これまでは8086/80286系マシン用のWindows/286（ただし日本国内ではこれをMS-Windowsと呼んでいた）と80386マシン用のWindows/386の2種類のWindowsが存在した。Windows 3.0でこれらは一本化された。かわりに，WindowsはCPUとメモリの実装容量に応じて3つのモードを持つこととなった。「リアルモード」「スタンダードモード」「386エンハンストモード」である。

Windows 3.0を起動時，Windowsは最初にハードウェアのチェックを行う。CPUの種類とメモリ容量から判断して最適なモードで起動する。実際にどのモードで起動したかを調べるにはプログラムマネージャの「Help」メニューから「About Program Manager」という項目を選択すればよい。それでは各モードごとにどのような違いがあるかを見ていくことにしよう。

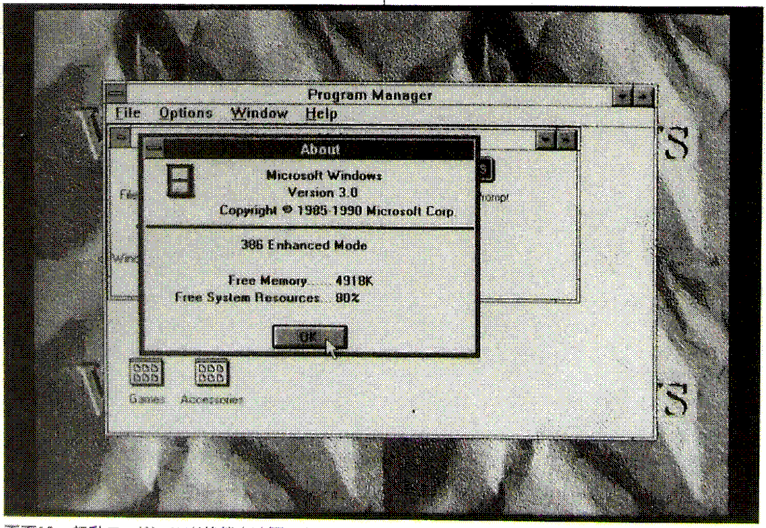
リアルモード

CPUが8086/8088の場合，80286以上のCPUでメモリが1MB未満の場合にはWindowsはリアルモードで起動する。初めからリアルモードを指定して起動したい場合は，WIN /Rとオプションを付ける。

リアルモードではWindows 2.1以前と同じ方式でメモリを管理する。多くのメモリを必要とするアプリケーションを動かすには，EMSメモリの利用が必須となる。

ただしこのモードではWindows 3.0の機能を十分に利用することができない。現に米国のWindows 3.0対応のアプリケーションの一部にはリアルモードでは動作しないと明確に表示されているものもある。つまり，見かけはWindows 3.0だが動作の仕方は2.X以前のWindowsのままである。リアルモードは旧バージョン用のWindowsアプリケーションを動かすためのモードといえる。

リアルモードではDOSアプリケーションを動作させることができる。従来のWindowsはメインメモリにWindowsとWindowsアプリケーションを置き，残りの空き領域でMS-DOSアプリケーションを起動しようとしたため，実際に動作するDOSアプリケーションはほとんどなかった。Windows 3.0では，MS-DOSのアプリケーションを起動するときにWindows自体のメモリイメージをディスクにスワップアウトする。Windowsに戻るときには，MS-DOSアプリケーションのメモリイメージをディスクにスワップアウトする。メモリをお互い食い合うことなく利用できるため，ほとんどのMS-DOSアプリケーションがメモリ不足で動かないという事態はなくなるだろう。



画面10　起動モードとメモリ状態を確認できる

EMS
Expanded Memory Specification。Lotus, Intel, Microsoftの3社によって定められた規格であるため，LIMと呼ばれることもある。現在のバージョンは4.0。メモリ空間の一部にページフレームという一種の窓を設定し，この窓の中にEMS対応拡張メモリの一部をマッピングすることによって8086 CPUのアドレス範囲外のメモリにアクセスする方式。

Windowsプログラミング用語
アナイソトロピックモード（Anisotropic mode）　Windowsのマッピングモードのひとつ。ウィンドウのサイズ変更時に画像比とスケールの両方の変更ができる。

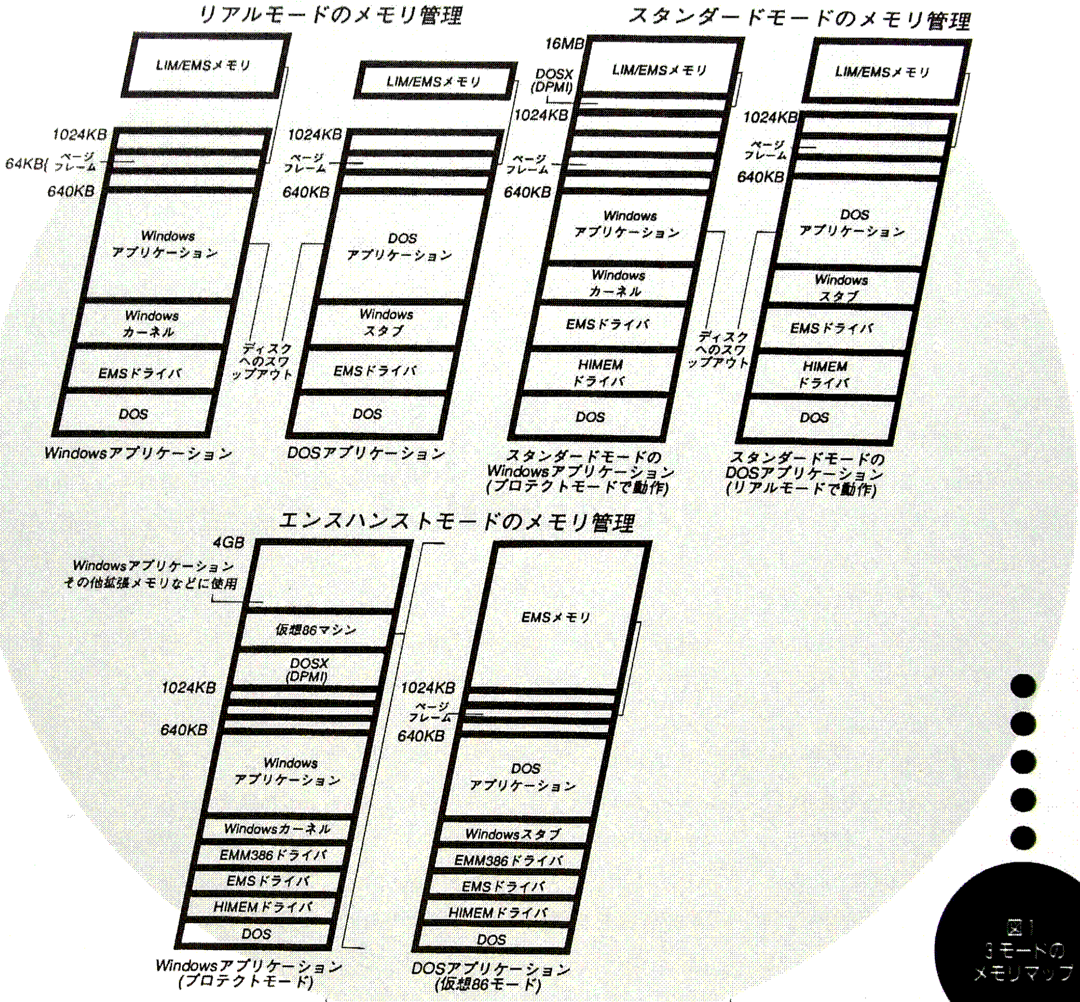


図1 3モードのメモリマップ

スタンダードモード

80286CPUでメモリ容量が1MB以上ある場合，80386CPUであってもメモリ容量が2MBに満たない場合にはWindowsはスタンダードモードで起動する．初めからスタンダードモードを指定して起動したい場合は，WIN /Sか/2オプションを付ける．

スタンダードモードで起動すると，Windowsは1MB以上16MBまでの拡張メモリ（Exended Memory）に直接アクセスすることができる．このモードを使ってプログラムを作成すれば，これまでのようなメモリ不足は一挙に解消されることとなる．その結果，スタンダードモードではメモリセグメントの破棄や再ロードなどの回数が減って，Windowsプログラム自体の実行の効率もアップする．

ただし，従来のWindowsアプリケーションは必ずしもこのモードで動くとは限らない．とくに実アドレスモードを前提として作られているようなWindowsアプリケーションは動作しないと思ってよいだろう．

スタンダードモードでもリアルモードと同様の方式でDOSアプリケーションを実行することが可能である．

XMS

Extended Memory Specification．80286以上のCPUで1MBを超えるメモリをアクセスするための方式．1MBを超える約64KBの領域（HMA）にアクセスする機能と1MBを超える領域のメモリブロック（EMB）にアクセスする機能がある．

386エンハンストモード

CPUが80386/386SX/486でメモリ容量が2MB以上のマシンでは，Windowsは386エンハンストモードで起動する．初めから386エンハンストモードで起動したい場合は，WIN /Eか/3オプションを付ける．この方法を使えばメモリ容量が2MBに満たないマシンでも386エンハンストモードで起動できるが，ディスクへのスワップアウトが頻繁に起きるので「使える」状態にはならない．

386エンハンストモードで起動すると，Windowsアプリケーションは最大で4GBまでの拡張メモリに直接アクセスすることができる．さらに，このモードではハードディスクを使った仮想記憶が実現される．仮想記憶とは実装メモリ容量よりも大きなメモリ空間を提供するためのOSの仕組みのことだ．Windows 3.0ではメモリ中のアクセス頻度の低い領域を4KBというページ単位でディスクにスワップする．スワップしたメモリ内のデータにアクセスするときには別の領域をスワップしてメモリと交換する．これらの処理はハードウェアによって制御されるので，アプリケーション側からはハードディスクが拡張されたメモリであるかのように見えるというわけだ．

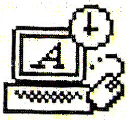
また，このモードは80386の仮想86マシン機能を利用できる．これによってDOSアプリケーションについてもWindows内で実行することが可能となった．リアルモードやスタンダードモードでDOSアプリケーションを起動しようとすると，たったひとつのDOSアプリケーションがフルスクリーンを占有してしまうが，386エンハンストモードでは複数のDOSアプリケーションをそれぞれウィンドウ表示できる．アプリケーション相互のカットアンドペーストも簡単だし，プログラム相互の切替えは386のハードウェア上の機構によって行われるため非常に高速である．DOSアプリケーションはそれぞれが仮想的な86マシン上で動作しているからEMS/XMSの容量を設定することもできる．

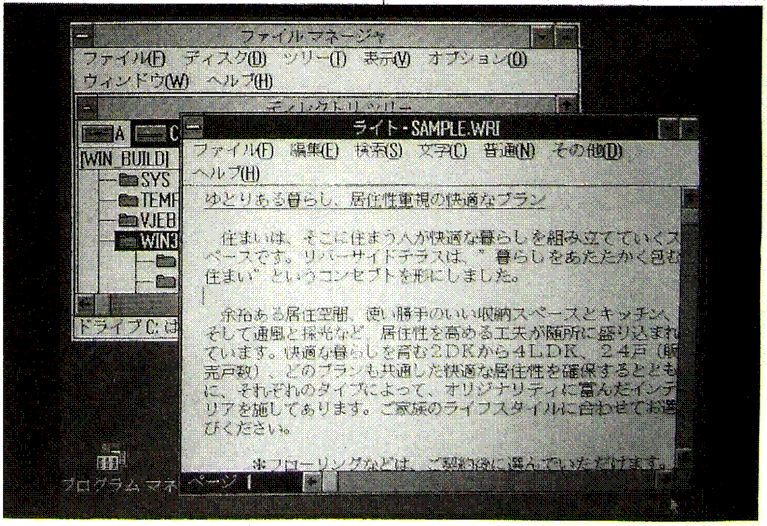
気になる日本語Windows 3.0とプログラム開発環境

MS-DOSパソコン初の本格的なGUIを備えたWindows 3.0の日本語版は91年春から各パソコンメーカーから発表される見通しだ．しかし，PCといえばIBM PCとその互換機しかない米国の市場と，さまざまな規格のパソコンがある日本とではWindowsを受け入れる土壌が異なる．Windows 3.0は果して日本でもエンドユーザーに利益をもたらすプラットフォームとなりえるだろうか？

Windows 3.0がそのメリットを十分に発揮するのはやはり386エンハンストモードである．CPUが386でメモリが2MB程度（実際にはパソコンの機種によって若干の差があるようだ）あればWindowsは386エンハンストモードで起動する．しかし，このマシン構成で起動できるMS-DOSアプリケーションはせいぜい1本がいいところだ．マイクロソフト社ではWindows 3.0を使うには4MB以上のメモリがあるのがのぞましいといっている．日本ではこれにかな漢字変換が加わる．アルファベットよりもはるかに多くを必要とする漢字フォントの問題もある．日本語版のWindowsはさらに多くのメモリを要求することになるだろう．

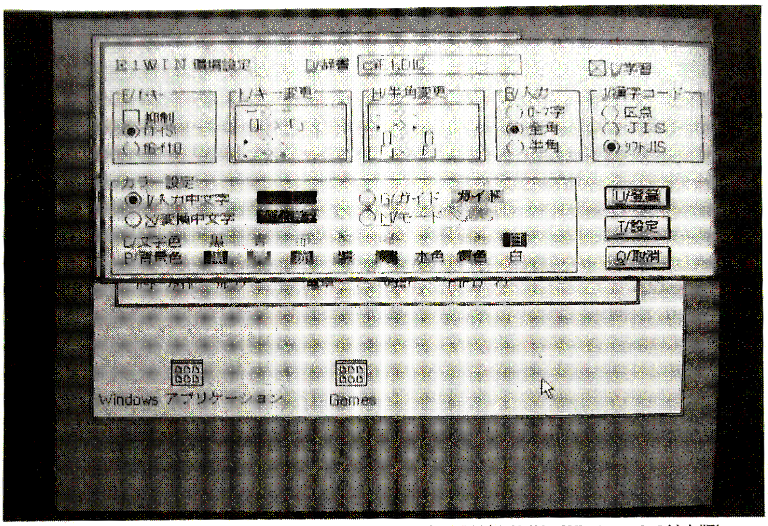
このところ日本国内でも80386マシンがかなり普及してきたが，まだまだ主流は80286以下メモリ1MB未満のマシンである．これらのマシンではWindows 3.0のリアルモードしか利用できない．Windows 3.0の普及は，魅力的なアプリケーションがどれだけ出てくるかにかかっているだろう．





画面11　まもなく登場する日本語Windows 3.0

Windowsプログラミング用語
インスタンス（instances（of program））　同一のプログラムを複数同時に実行した場合，各タスクに展開されたプログラムをインスタンスという．



画面12　日本語環境ではWindows用の日本語入力ソフトが必要だ(E1WINのWindows 3.0対応版)

日本語版Windowsアプリケーションの開発

Windows普及の鍵となるアプリケーションプログラムだが，Windows 2.1に対応するWindowsアプリケーションは決して多いとはいえなかった．その理由のひとつには，Windowsでのアプリケーション開発のむずかしさがあげられる．Windowsという OSを理解し，Windows用のアプリケーションが書けるようになるまでには，熟練したプログラマでもある程度の勉強は必要だし，しかもWindows 2.XではMS-DOS上のメモリの制限内にアプリケーションプログラムを収めなければならない．とくに日本ではかな漢字変換でメモリの制限はさらにキツくなるし，Windowsユーザーも少なかったからなおさらである．

しかし，Windows 3.0はソフトウェアハウスやプログラマにとって注目すべき環境に変貌したといえるだろう．新しいWindowsでは，特別なDOSエクステンダーを組み込まなくてもプロテクトメモリにアクセスできる．複雑なEMSのハンドリングは必要ない(リアルモードでは別だが)．また，洗練されたユーザーインターフェースを利用できることも魅力だ．

開発環境も2.Xのときに比べるとずいぶん改善された．一番大きなポイントはコードビューが使えるものになったことだろう．IBM PC用のWindows SDKではPCにインターフェースボードを入れてそこにモノクロディスクプレイをつないで，デバッグ画面を表示する方式だが，日本語版SDKでは画面切替え方式でコードビュー(CVW)が使えるようになるそうだ．

プログラマの層は薄い　米国製のアプリケーションがどっと流入？

そうはいってもWindowsに対応したアプリケーションプログラムを作成するには，MS-DOS用に書かれたものをちょっと変更するといった程度ではすまない．Windowsアプリケーションの開発はWindows用のAPI (Application Programming Interface)を使って行う．これまでMS-DOS上でいくらかプログラミングの経験がある人でも最初はかなりやりにくいだろう．

このような状況にあって，ソフトウェアメーカーのいくつかは新しく自社でプログラムを開発するのではなく，米国ですでに実績のあるWindows用アプリケーションを日本語化するという道を選択した．米国でのWindowsアプリケーションも，マッキントッシュ版からの移植が多いことを考えると，新しいOSの経験を積むには最適の手段かもしれない．

CHART

米国のWindowsアプリケーションを移植／販売しているおもな会社（含む予定）

会社名	主なアプリケーション名
マイクロソフト	Excel
サムシンググッド	Aldus PageMaker
データポット	Actor
ジィスクソフトウェア	Mictograph Designerほか
アイネス	Pixie（日本語名「ぴっくす」）
三菱商事	SQLWindows
エム・ピー・テクノロジー	Guide

Windowsプログラミング用語
ウィンドウクラス（Window Class）　データ構造や操作，機能の同じウィンドウをグループ分けしクラスと呼ぶ．全てのウィンドウはクラスで分類される．

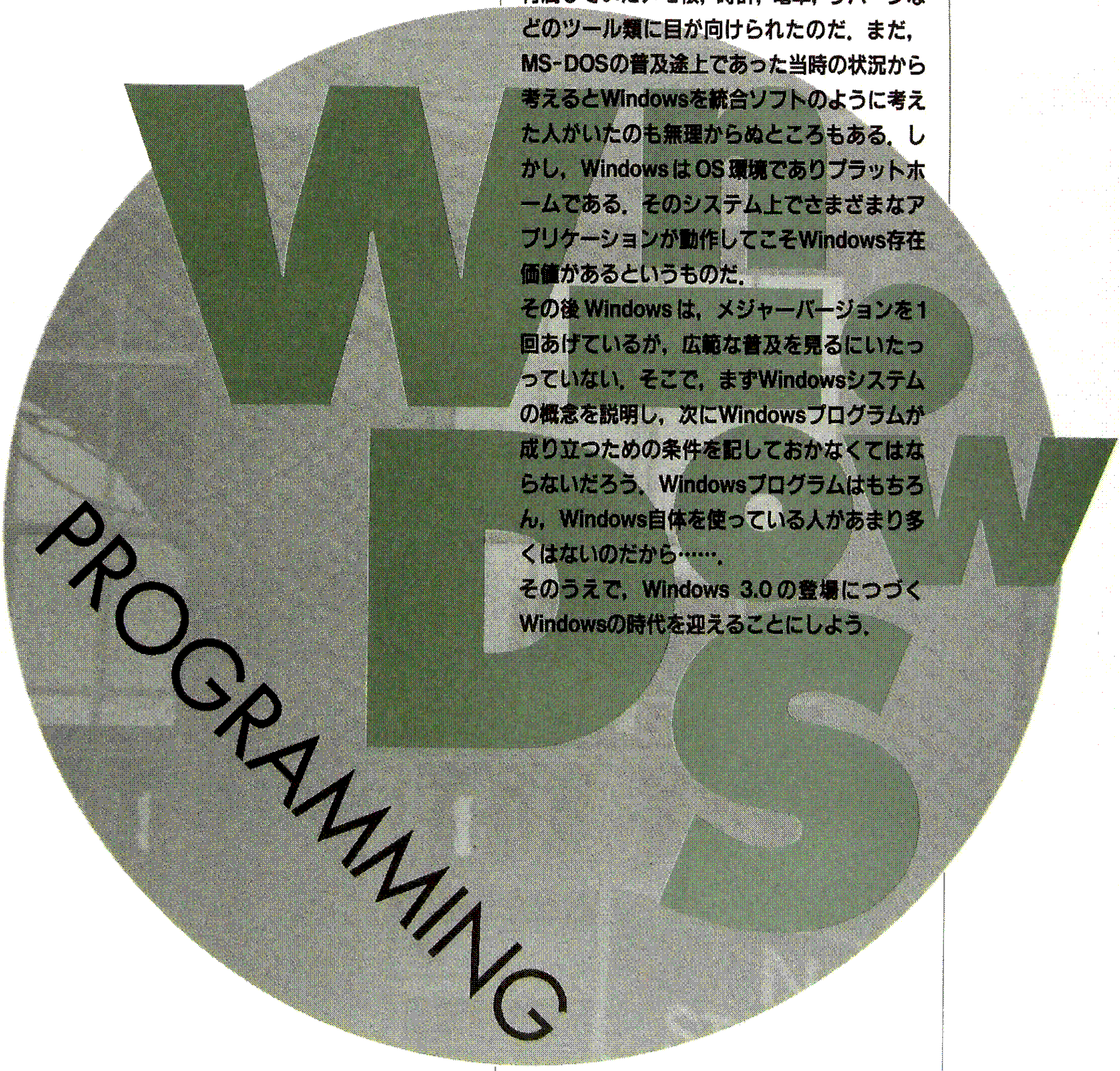
Windows
プログラミング講座
1

Windowsシステムとプログラムの条件

1986年にMS-Windowsのバージョン1が日本でリリースされたころは，MS-DOS上のGUI環境というよりも，グラフィックベースの統合デスクトップツールが登場したという受け取りかたをされたようだ．つまり，Windowsに付属していたメモ帳，時計，電卓，リバーシなどのツール類に目が向けられたのだ．まだ，MS-DOSの普及途上であった当時の状況から考えるとWindowsを統合ソフトのように考えた人がいたのも無理からぬところもある．しかし，Windows は OS 環境でありプラットホームである．そのシステム上でさまざまなアプリケーションが動作してこそWindows存在価値があるというものだ．

その後 Windows は，メジャーバージョンを1回あげているが，広範な普及を見るにいたっていない．そこで，まずWindowsシステムの概念を説明し，次にWindowsプログラムが成り立つための条件を記しておかなくてはならないだろう．Windowsプログラムはもちろん，Windows自体を使っている人があまり多くはないのだから……．

そのうえで，Windows 3.0 の登場につづくWindowsの時代を迎えることにしよう．



1 表裏一体のGUIとデバイス独立

MS-Windowsにふれたことのない多くの読者のために，Windowsのプロフィールを紹介することにしよう．MS-DOSのコマンドを使うときにその書式に沿って入力しなければならないように，MS-Windowsにもさまざまな決まりごとがある．こうした決まりごとによって，さまざまなWindowsソフトに操作の共通性が実現されている．つまり，Windowsのプログラムを組むときにも知っておかなくてはならないことである．なによりも，自分の作るプログラムがどういう規則の中で，どういった動きをするのかをつかんでおく必要があるだろう．

GUIの前面 「絵」で構成される画面

MS-Windowsのグラフィックユーザーインターフェースは，画面に表示される「絵」によって「使う人」がより簡単にパーソナルコンピュータを操作できるようにすることをめざしている．ここでいう「絵」とは，アイコンなどの「絵文字」や「図形」以外に，グラフィックスで表示される文字も含まれる．絵で構成された画面には，さまざまな機能が埋め込まれていて，それぞれ名称がつけられている．Windowsの特色である操作の共通性は，表面的にはこうしたウィンドウの機能統一に始まるといえる．Windows上でさまざまなアプリケーションを動作させても，その操作性はおおむね似ているという環境が作られている．図1にWindows 2.XとWindows 3.0を並べて各部分の名称を記しておこう．

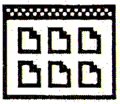


図1 Windows画面の構成要素

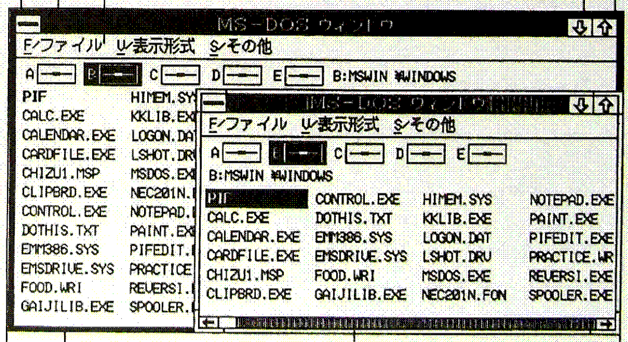
②タイトルバー
プログラムのタイトルを表示するエリア．中央にウィンドウタイトルが表示される．

③メニューバー
このエリアにプルダウンメニューが納められている．メニュー内容はプログラムによって異なる．

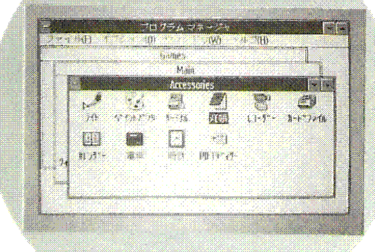
④アイコン表示ボックス
ここをクリックするとそのプログラムウィンドウはアイコン表示になる．

⑤フルスクリーン表示ボックス
ここをクリックするとフルスクリーン表示される．同時にこのボックスは，ノーマル表示ボックスに変更される．

①コントロールメニューボックス
各Windowsアプリケーションがもっている．ウィンドウの大きさや位置の変更，プログラムの終了などを行う．



Windows 3.0のプログラムマネージャでは，グループアイコンやグループウィンドウを使って操作する．



⑥ウィンドウ境界線
ひとつのウィンドウを囲む境界線

⑦スクロールバー
データをウィンドウに表示しきれない場合，スクロールバーが出る．スクロールボックスまたは矢印をドラッグしたりクリックすることでデータをスクロールさせることができる．

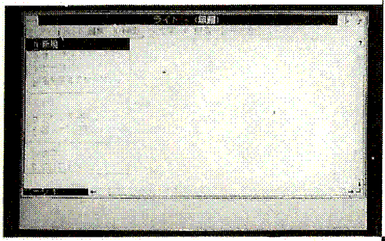
Windows上のアプリの動き
必要に応じて登場するWindowsのメンバー

Windowsに付属しているアプリケーションから「ライト」を起動してみよう。「ファイル」メニューの「オープン...」コマンドは「...」がついている（**画面 1**）。これは，このコマンドを選択すると子ウィンドウが表示されるという意味だ．この場合は，子ウィンドウのひとつであるダイアログボックスが現れて読み込むファイルを選択できるようになる（**画面 2**）．ダイアログボックスはユーザーが入力や選択を必要とする場合などに使われるWindowsの構成要素で，Windows開発キット（SDK）には，ダイアログエディタというツールも用意されている．

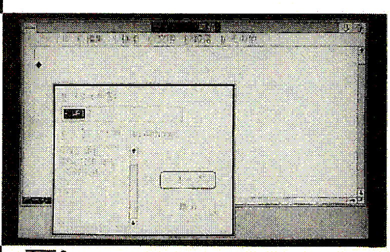
次に，「ペイント」を起動する．ここで，ブラシを選択して簡単な絵を描いてみる（**画面 3**）．範囲指定をするツールを選択して，絵を含むように指定する．「編集」メニューから「コピー」コマンドを実行する．「ライト」に移動して「編集」メニューから「ペースト」コマンドを実行する．これで，ペイントに描いた絵は「ライト」に貼りこまれる（**画面 4**）．

ここで，「MS-DOSウィンドウ」に移動して「clipbrd. exe」を起動すると，先ほどの絵が表示される（**画面 5**）．このようにクリップボードは，異なったWindowsアプリケーションの間でデータをやりとりする場合の仲介役を果たす．ペーストされるデータの内容はクリップボードアプリケーションを開くとみることができる．

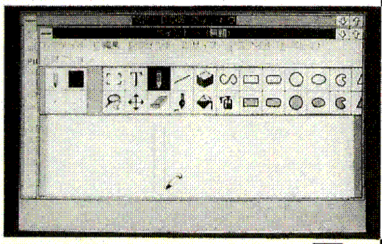
このようにWindowsでは，必要に応じてWindowsを構成するメンバーが登場する．



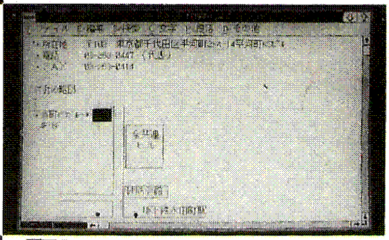
画面 1
「ライト」を起動する



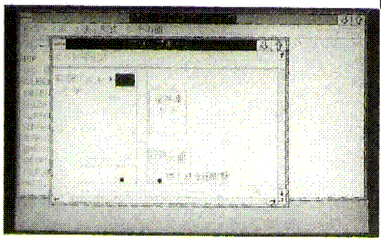
画面 2
ダイアログボックスが現れて，ファイルが表示される



画面 3
ペイントで絵を描き，範囲指定してからコピーする



画面 4
ライトに移動してペーストするとペイントで描いた絵が貼りこまれる



画面 5
編集対照となったデータはクリップボードで見ることができる

グラフィックユーザーインターフェース
「絵」で構成することの意味

Windowsでは画面全体が「絵」であり，それをグラフィックユーザーインタフェース（GUI）ということができる．ところが，GUIという言葉がひとり歩きしているきらいがあって，「絵」で画面を構成することによって生じるもろもろの約束ごとがグラフィックユーザーインターフェースを成り立たせていることは案外認識されていない．プログラミングという観点からみた場合，「もろもろの約束ごと」のほうがむしろ現実的な問題になるはずだ．

マイクロソフトの古川社長は，セミ

Windowsプログラミング用語
キャプションバー（Caption bars）　各ウィンドウの表題（＝キャプション）を表示する領域で，タイトルバーともいう．

ナーなどの席で「CUIからGUI」への移行を，「活字から写植への」変化にたとえて話している．CUIの世界では，文字キャラクタは鉛のかたまりである活字と同様に，１文字という単位でしか取り扱うことができない．40字×25行というテキスト画面は，箱に入った1000個の活字と同じことだ．それがGUIになると，たとえば640×400ドットの画素によって構成される「絵」なのである．テキスト画面の文字を固い活字とし，グラフィック画面の文字をレンズによって拡大・縮小・変型などが可能な写植文字にたとえるのは的確である．

画面を「絵」にするという処理こそが，デバイスから独立したWindowsのプログラム環境を実現しているのである．テキスト画面とグラフィック画面を持っているコンピュータでGUIとCUIの差を見てみよう．日本語Windowsの場合，通常はパソコン本体がもっている漢字ROMのフォントパターンを使って漢字を表示するため，CUIでもGUIでも同じ16ドットなり24ドットのフォントパターンが表示されるから，表面的に違いはない．

同じフォントパターンを表示しても，システムとして考えるとそこには本質的な違いがある．40字×25行と640×400ドットは，基本的に異なるのである．漢字の場合は，**図２**のような経路をたどって，画面に文字が表示される．

Windowsシステムでは，いったん漢字ROMのフォントパターンをとってからそれをビットマップに展開して，ふたたびハードウェアに送り，それがディスプレイ表示される．そのときに，表示デバイスの機能に沿ってビットマップを展開するのはWindowsシステムの役割だ．アプリケーションは表示デバイスに関する情報を気にしなくてもいいことになる．それはプリンタに関しても同じだ．Windowsシステムでは，プリンタデバイスをWindowsが管理する．ユーザーは，適切なプリンタドライバをWindowsに指示するだけでいい．これを実現するためにWindowsは，デバイス管理のかなりの部分をDOSにかわって行っている．これを説明するためには，ハードウェアとDOS，Windows，アプリケーションの関係を見る必要がある．

DOS上とWindows上のアプリケーションの違いハードウェア制御

DOSアプリケーションは，**図３**のような概念のシステム上で動作することになっている．DOSのファンクションを使ってプログラムされた行儀のよいアプリケーションは，ハードウェアの相違を吸収してくれるDOS上で動作する，ということがいわば理想像として語られている．現実がそうなっていないことはご存知の通りである．DOSの機能は，ディスクの入出力以外をほとんどサポートしていないからである．DOS上のアプリケーションプログラムは，ディスクの入出力に関してはDOSの提供する機能を使うが，ほかのデバイスに対しては必要に応じてROM BIOSやハードウェアを直接アクセスしている．

これに対して，Windows上ではシステム自体がディスク以外のデバイスをアプリケーションに代わって制御している．Windowsアプリケーションは，それぞれがプリンタやディスプレ

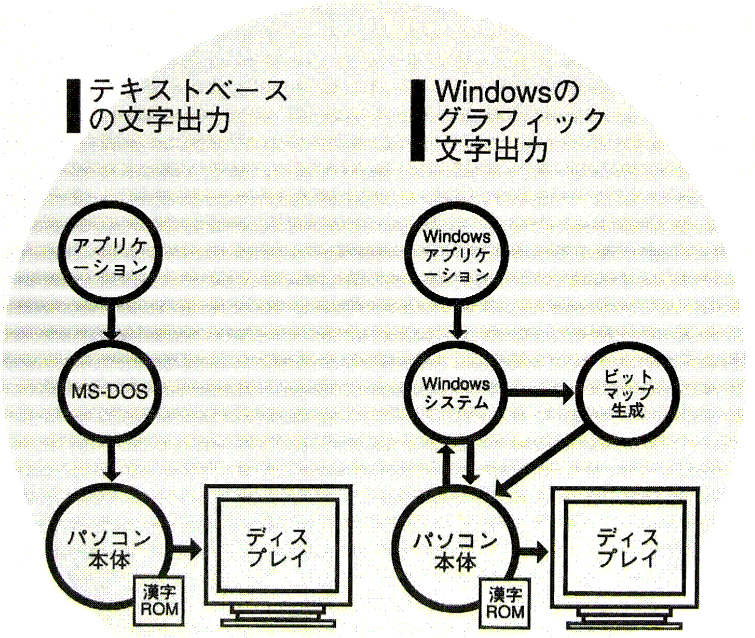


図２
画面表示の違い

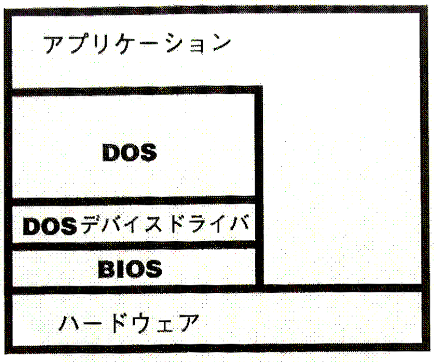


図3 MS-DOSのシステム概念

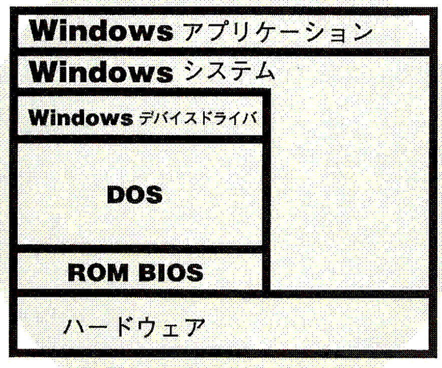


図4 Windowsのシステム概念

イなどのデバイスを制御するコードを持つ必要はない．それぞれの位置関係は図4のようになる．

Windowsシステムを構成するファイル
システム状況に応じたファイルセット

Windowsシステムは中心となるKERNEL. EXE, GDI. EXEそしてUSER. EXEの3ファイルから構成されている．Windows 2. Xでは，この3ファイルはインストール時にWIN200. BINとWIN200. OVLとしてコピーされる．Windows 3. 0では，これらのファイルはサブディレクトリSYSTEMにコピーされ，起動プログラムであるWIN. COMがシステム状況をチェックし，それに応じて必要なファイルセットを起動するようになっている．つまり，80286や80386CPUのマシンでは動作するシステムファイルが異なる．起動の流れとファイルセットの関係は図5のようになる．

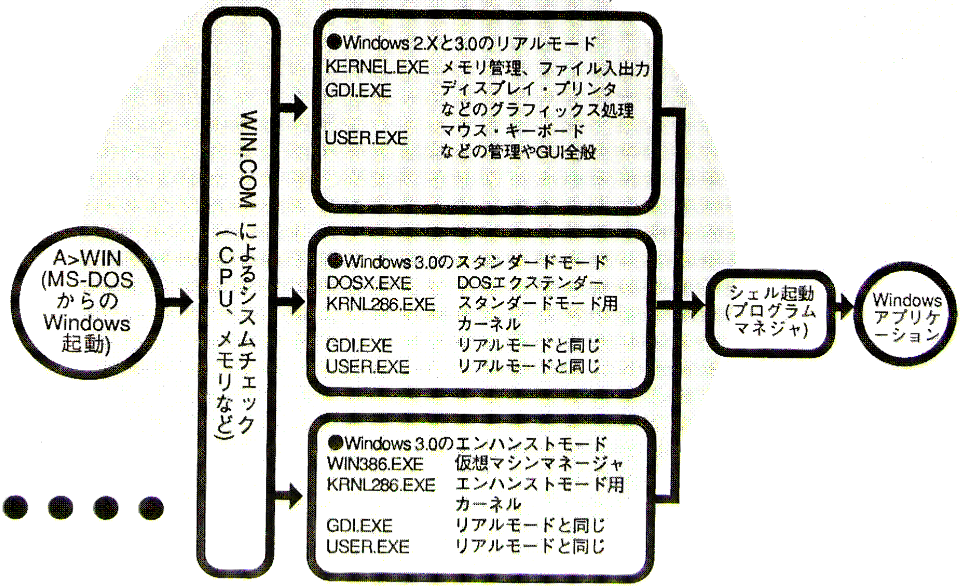


図5 Windowsの起動とプログラム構成

Windowsプログラミング用語
クライアントエリア（Client area） プログラムがGDIを介して自由に描画や制御できるウィンドウ内の領域のこと。

2 Windowsプログラミングの条件
Windowsアプリケーションを作成するために

Windowsアプリケーションの作成はMS-DOS上で動作するアプリケーションの作成とはかなり趣が異なる．その理由はWindows上でのアプリケーションはWindowsという環境の上で他のアプリケーションと共存しつつ動作しなければならないことにある．Windowsアプリケーションの構造についての説明は本特集の後半に譲ることとして，ここではMS-DOS上のアプリケーションとの比較を考えながらWindowsアプリケーションを作成するために必要な「条件」を考えてみよう．

Windowsプログラミングの条件 その1 これだけは必要，ハードウェアとソフトウェア

Windows上で動作するアプリケーションを開発するには**表1**のようなハードウェア，ソフトウェアが必要である．

ハードウェアについてはここに上げた条件は最低限のラインだと考えて欲しい．Windowsはシステムそのものが大きいので，その上でさらにアプリケーションを作成，デバッグするにはメモリ容量は多いに越したことはない．またWindows3.0でプロテクトメモリを使用するプログラムを作成するときには，開発マシンにプロテクトメモリが必要なことはいうまでもない．

動作させるWindowsのバージョンによって違う点は，デバッグ用のターミナルが必要かどうかだ．Windows2.XまではRS-232Cポートに端末をつないで，そこへデバッグ用のメッセージを出力していたがWindows3.0ではモノクロディスプレイを1台接続し，そこへWindows用コードビューデバッガの出力を表示できるようになった（IBM-PCとその互換機の場合）．日本語版のWindows3.0では，1台のパソコンでデバッグ用の画面に切り替えることができるようになるといわれている．

ソフトウェアについてはCまたはアセンブラが必要である．現在のところWindowsに対応したCの処理系はMS-Cだけ，アセンブラはMASMだけだと考えてもらって差し支えないだろう．さらに，Windowsソフトウェア開発キット（SDK）が必要になる．この開発キットにはWindowsアプリケーション作成のためのライブラリや各種デバッグツールなどが含まれる．

Windowsプログラミングの条件 その2 Windows APIを使え

Windowsは疑似マルチタスク環境である．ひとつのアプリケーションの処理の途中でもこれを終了せずに別のアプリケーションが動く（もっともこの間は元のアプリケーションは「停止」しているので，厳密な意味ではマルチタスクとはいえないが）．このような環境で動作するアプリケーションが，シングルタスクであるMS-DOS上のプログラムのように，CPUやメモリ，ファイル，プリンタなどを独占してしまったら，他のアプリケーションは動けなくなってしまう．Windowsアプリケーションでは制御をWindowsシステムそのものにまかせることによって共有を実現している．

実際のメモリやハードウェアに関わる処理を行うときは，特別なプログラムインターフェースを使う．これをWindows API（Application Programming Interface）という．

Cの標準入出力は使えない！ 専用のWindows関数でプログラミング

WindowsではCプログラマにはおなじみのprintf()関数などのC言語の標準入出力関数は使えない．かわりにWindows専用の関数を多用する．Windowsの関数にはウィンドウを作成，移動，変更したり，ダイアログボックスを表示したりといったWindowsの基本的な要素となる関数や，グラフィックを使った描画に関する関数，タスク管理やメモリ管理を行う関数などさまざまな関数がある．Windowsのアプリケーションプログラム

Windowsプログラミング用語
クリップボード（Clipboard） プログラム間のデータの転送など，メモリブロックの交換を行うWindows関数．同名のWindowsアプリケーションもある．

はこれらのWindows関数を使って記述しなければならない．

ポインタを保持してもムダ！
Windowsのメモリ管理

640KBというMS-DOSのメモリ上で複数のアプリケーションを動かすには，MS-DOS上のように，アプリケーションに自由にメモリを使用させるわけにはいかない．そこでWindowsシステムがつねに現在の状態を把握してメモリの適切な配置を決定する．ある時点でメモリ上のどこにアプリケーションがロードされているのか，プログラマはその正確な位置を知ることはできない．したがって，Windowsアプリケーションを作成するときに関数やデータのポインタを保存しておいても，次に利用するときにそれが有効であるという保証はまったくない．

Windows上でメモリにアクセスするときには「ハンドル」を用いる．ハンドルは実際にはunsigned int型の値で，実際のメモリにアクセスするためのインデックスの役割を果たす．Windowsはメモリの物理的な番地とハンドルとの対照表を持っていて，メモリを再配置するたびにこの対照表を更新する．アプリケーションプログラムがメモリにアクセスしたいときは，このハンドルの値を指定すればWindowsシステムが実際の番地に変換してくれるというしくみになっている．

入力はWindowsシステムにまかせろ！

Windowsアプリケーションはマルチタスク環境の元で動作するのが前提だからハードウェアを直接制御することは許されない．キーボードやマウス，タイマからの入力はすべてWindowsシステムがシステムキューという形で受け取る．Windowsシステムはその入力がどのアプリケーションに対するものであるかによって，各アプリケーションごとに存在するアプリケーションキューにコピーする．

アプリケーションの側では直接ハードウェアから入力を受け取るかわりに，入力をキューから読み込む．入力の情報は「メッセージ」という特別なフォーマットで渡される．「キーが押された」(WM_KEYDOWNメッセージ)，「マウスの左ボタンがクリックされた」(WM_LBUTTONDOWN)というぐあいだ．

Windowsプログラミングの条件 その3
リソース定義を生かせ

Windowsの画面表示にはアイコン，カーソル，メニュー，ダイアログボックスなどの独自の要素がある．Windowsプログラミングではこれらをまとめて「リソース」と呼ぶ．Windowsアプリケーションでは，リソース

CHART

Windowsアプリケーション開発に必要なハードウェア&ソフトウェア

[Windows 2.Xの場合]
- MS-Windowsが動作可能なパーソナルコンピュータ　1台
  (メインメモリ640KB以上，1台以上のフロッピーディスクドライブとハードディスクドライブ，モノクロまたはカラーディスプレイ，キーボード付)
- ダムターミナルまたはパーソナルコンピュータ（デバッグ用）　1台
- MS-DOS Ver3.0以上
- MS-Windows 2.X
- MS-Cコンパイラ Ver4.0以上またはマクロアセンブラMASM Ver4.0以上
- MS-Windows ソフトウェア開発キット（SDK）Ver2.0

[Windows 3.0の場合]
- Intel社の80286，80386（386SX含む）以上のCPUを備えたパーソナルコンピュータ　1台
  (メインメモリ640KB＋拡張メモリ384KB以上，1台以上のフロッピーディスクドライブとハードディスクドライブ，モノクロまたはカラーディスプレイ，キーボード，マウス付)
- モノクロディスプレイとMDA（Monochrome Display Adapter）カード（IBM PC互換機でデバッグする場合）
- MS-DOS Ver3.1以上
- Windows 3.0
- MS-Cコンパイラ Ver5.1以上またはマクロアセンブラMASM Ver5.1以上
- Windows ソフトウェア開発キット（SDK）Ver3.0

Windowsプログラミング用語
グラフィックデバイスインターフェース（GDI）　デバイスとプログラムの間のインターフェースで，プログラムのビジュアルな情報処理を支援するライブラリ．

だけをまとめて１つのファイルに定義する．これをリソース定義ファイルという．SDKにはリソース定義を作成するためのさまざまなリソースエディタ（ダイアログボックスエディタ，アイコンエディタ，フォントエディタ）がついてくるのでおもなものはこれを使って作成する．

Windowsアプリケーションの特徴のひとつは，リソースの取り扱いがアプリケーションの手続き部分と独立していることだ．アプリケーションの手続き部分はCまたはアセンブラでコーディングし，コンパイルまたはアセンブル後リンクして実行可能形式のファイルとなる．リソース定義ファイルはリソースコンパイラ（RC.EXE）というリソース専用のコンパイラを使ってでコンパイルし，最後に実行可能形式のファイル（*.EXE）に付加される．

したがって，ダイアログボックスの大きさを変更したり，カーソルの形状を変更したりするだけならいちいちソースプログラムをリコンパイルする必要がない．リソース定義ファイルのみをコンパイルして実行可能ファイルに付け加えればよいわけだ．リソースの作成手続きが独立していることによって，Windowsアプリケーションでは比較的簡単にユーザーインターフェースの構築，修正ができる．

COLUMN

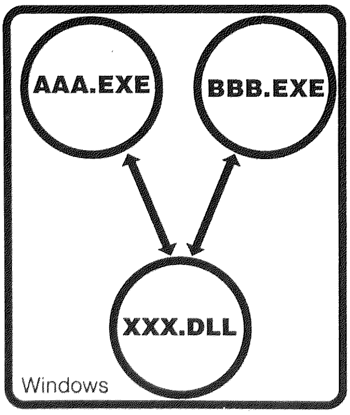
マルチタスク環境を実現するWindowsの「技」

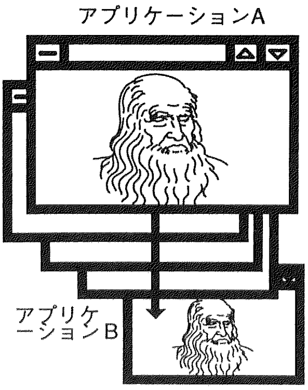
DLL

Windows関数のライブラリは通常のC言語のライブラリとは異なる特殊な形式をとっている．これはダイナミックリンクライブラリ（DLL）と呼ばれる．DOS上でのアプリケーションではプログラムを作成するときに必要なモジュールをすべてリンクしてしまうのに対して，Windowsライブラリは Windowsシステムがそのアプリケーションをロードするときに初めてリンクされる．同じアプリケーションが複数動作しているときは，ひとつのライブラリルーチンを共用できるので実行に必要なメモリ量が節約できるというメリットがある．

DDE

異なる２つのアプリケーションでデータを交換しようとする場合（たとえば一太郎にマルチプランの表を読み込むことを考えてもらえばわかるだろう），MS-DOS上ではいったんファイルを経由しなければならなかった．Windows上では異なるアプリケーションの間でもデータやメッセージを転送することができる．これを実現しているのが，ダイナミックデータエクスチェンジ（DDE）というプロトコルである．





Windowsプログラミング用語
グローバルメモリ（Global memory）　MS-DOSがWindowsをロードした先頭から物理メモリの最後のアドレスまでをグローバルメモリという．

Windows プログラミング講座 2

Windows 3.0+MS-C 6.0による開発環境構築法

永井知己

Windows 3.0の開発をしたいのだがさてどうしたものか？ どんなハード・ソフトウェアが必要なのだろう……．そう思っている人は少なくないはずだ．本稿はそんな人のための環境構築法となっている．まだ日本語版のWindows 3.0も，その開発キットもリリースされていない段階で書いたものだから，すべて米国版を使って行っているが，これから日本語版のWindows 3.0で開発を行おうという読者の参考になるはずだ．また，Windows 3.0だけでなく，Windows全般の開発環境の解説として読むこともできる．

さて，今回使ったハードウェアはPS/55Zだ．55ZのPS/2モードで１から環境を作り上げてみた．DOS4.0だけがインストールされた55Zに，Windows 3.0とMS-C 6.0，それにWindowsの開発キットによる開発環境を築いた．私は，Windows 2.1にMS-C 5.1という環境で開発するのには慣れていたが，実のところWindows 3.0用のソフトの開発も，MS-C 6.0でのコンパイルもやったことがない．心の隅に不安を覚えながらも，どうせ似たようなものだろうと始めのうちは高をくくっていた．まさかその後で地獄をみるとは思いもせずに．そう，この記事はみなさんがインストールに失敗しないためのガイドなのだ．

WINDOWS PROGRAMMING

Windowsプログラムに必要なもの

Windowsのソフトを作ろうと思ったときまず困るのが，何が必要なのかということだ．どれくらいのハードがあればいいか，どんなソフトが必要かなど，一度やってみなければわからないことはいがいに多い．そこで今回実際に使用した機器，ソフトウェア構成について説明しよう．

ハードウェア PS/55Zレベルは最低限必要

まず，PS/55Zについてふれておこう．PS/55ZのCPUは80386SXの16MHz，2MBのRAMと30MBのハードディスクが標準装備されている．Windowsの開発には必要最低限の環境だと思う．80386CPUはWindowsをエンハンストモードで使用するのには不可欠だ．MS-C 6.0はキャラクタベースのプログラムである．エンハンストモードでは，DOS上でしか動かないMS-Cと，Windowsをうまく融合させることができる．Windows上からパッとDOSに移り，開発した後にサッとWindowsに再びもどって実行する．2MBのRAMはエンハンストモードの最低容量であり，やはり最低限といえよう．また，イライラしない程度の使用には，16MHzのクロックはぜひとも欲しい．ハードディスクは30MBだが，全インストールが済んだときの空きエリアが約10MBだったことを考えると，実用にはつらいかもしれない．PS/55Zは日本IBMの製品であるが，今回は英語モードしか使用しない．したがって同様のスペックをもつIBM互換機なら何でもいいだろう．

ソフトウェア 英語版を使用

次はソフトを紹介する．使用するのはWindows 3.0，MS-C 6.0，Windows開発キット3.0（Software Deveropment Kit：以下SDK）の3つだ．アセンブラが使える人はMASMも用意しよう．Windows開発では使う機会がしばしばあるはずだ．実際，サンプルプログラムの中にはクロックなどアセンブラが必要なものがしばしばある．

この原稿を書いている現在，MS-C 6.0，Windows 3.0，SDK，いずれも日本語版は出荷されていない．そこで今回は英語版を手にいれて環境を構築することにした．現時点ではソフトウェアの入手については次のようにいえる．

第1に，マイクロソフト経由では購入に時間がかかること（マイクロソフトでも品不足の状態なのでこれはしかたないだろう．日本語版が出たときに期待しよう）．第2に，もし早急に必要ならアメリカのパソコン通信ネットの通信販売を利用すれば手に入る．申込みから数日で届けてくれる．定価＋α程度ですむ．

WindowsとCを高度に利用しよう

Windows 3.0は，プログラムの動作チェックに利用するほか，開発環境の中心として使うことができる．WindowsアプリケーションNの開発もWindows 3.0のマルチタスク環境のひとつとして実行してしまおう．MS-Cのツールをアイコンとして登録しておけば，これらのツールがまるでWindowsのアプリケーションの1つであるかのように作業することができる．また，電卓を使いにちょっとWindowsにもどることなど朝飯前だ．

大物はやはりMS-C 6.0である．5.1からの最大の飛躍はPWB（Programmers Work Bench）が標準装備されたことだろう．PWBは一見ただのエディタであるが，Quick Cを10倍パワーアップしたような統合環境を実現してくれる．Windowsも完全にサポートされており，初心者でも悩むことなくWindowsの開発をすることができる（実際には，私はこれに数時間を費やしてしまった．インストールマニュアルはよく読もう）．といっても，このPWBはWindows上で動くわけではないから，結局はDOSに戻る必要がある．

最後にWindowsSDK3.0．前バージョンのSDKにくらべ，関数がまた一段と増えた．500種類くらいはあるとおもう．でも，Windows上やPWB上でオンラインヘルプを使えば恐いものはない．PWB上では知りたい関数の上で［F1］キーを押すだけでよいので便利だ．

ソフトウェアのインストール

3つのソフトウェアを合計するとフロッピーは全部で20枚近い．見るだけで頭が痛くなってくること受け合いだ．インストールを失敗した日にはまたやり直し．これでは時間がいくらあっても足りない．そこで，初めてイン

ストールをする人は，私がこれから紹介する方法をお勧めする．私は3回目にやっとインストールに成功した（ただ，マニュアルをよく読んでいなかったためだが）．Windows 3.0よりも先にWindowsSDKをインストールしてしまったのが敗因であった．たっぷり数時間かかってしまった経験を生かし，以下に最短距離の方法を紹介する．

(1) Windowsを組み込む

まず，Windows 3.0をハードディスクにインストールする．「Windows 1」ディスクのsetup.exeを実行すれば，あとは指示に従うだけでよい．至極簡単な作業である．詳細はマニュアルを参照して欲しい．インストールが終わったらシステムをリセットしよう．

(2) MS-C 6.0を組み込む

「setup」ディスクのsetup.exeを実行．インストールが始まる．setupプログラムがいろいろと質問してくるはずだ．たとえば，

Copy documentation files [Y]:

という形式で尋ねてくる．[Y]というのはデフォルトの設定であり，ここでリターンキーを押すとYes，つまりファイルをコピーすることになり，[N]を押すとコピーしないことになる．すべての項目を入力し終えると，

Do you want to change any of the above options? [Y]:

と表示される．このときだけは修正が必要な場合は[Y]を，ばっちり完了というときは［N］を押す．

さて，数多くの質問に対してどんな設定にしていいか迷うと思う．いったい何を指示することがWindowsのコンパイルに必要なのだろうか．結論をいうと，すべてデフォルトのまま，つまりリターンキーを返し続けることが，いちばんの近道である．われわれの目的はあくまでもサンプルプログラムをコンパイルすることである．内容なんかわかってなくても，デフォルト指定のみで問題は起こらない．

(3) TOOLS.INIを作る

インストールが完了するとC600というディレクトリが作成されているはずだ．¥C600¥BIN¥をのぞいてみよう．するとTOOLS.PWBというファイルが目に入ってくる．これはPWBの環境設定ファイルだ．これをTOOLS.INIにRENAMEしよう．WindowsSDKのインストール時に必要となってくる．

(4) WindowsSDKを組み込む

開発の心臓部というべき，WindowsSDKをインストールする．このあたりで，へとへとになっていると思うが，我慢，我慢．「Development Tools」ディスクのinstall.exeを実行する．MS-Cのときのように

Do you want to change any of the above options? [Y]:

と質問してくるはずだ．やはりここでも，無難にデフォルトに従おう．

(5) CONFIG.SYSを設定する

今回は，Windows開発だけのためのCONFIG.SYSを作成した（LIST 1）．RAMが2MBしかないPS/55ZではWindowsをエンハンストモードで動かすのがぎりぎりである．したがって，キャッシュディスクのSMARTDRV.SYS（注1）は使用しないようにしている．HIMEM.SYSは，これがないとリアルモードになってしまうので削ってはならない．

(注1)SMARTDRV.SYS　ディスクキャッシュ用のデバイスドライバ．

(6) AUTOEXEC.BATを設定する

さて，ここまで済んだらしめたもの．AUTOEXEC.BATの設定が残っているのみである．Cドライブがハード

LIST 1

Windows 3.0開発用CONFIG.SYS

```
1:      files= 30
2:      buffers=25,8
3:      country=001,437 c:¥dos¥country.sys
4:      shell=c:¥ps2¥command.com /p /e:512
5:      device=c:¥himem.sys
6:      device=c:¥ps2¥ansi.sys /x
```

1,2: DOS4.0におけるfilesとbuffersの設定
3: 国別コードページの設定
4: command.comの設定．環境変数領域を512バイトと設定している
5: 拡張メモリドライバの設定．これがないとWindows 3.0がスタンダードモードやエンハンストモードで動作しない
6: エスケープシーケンスドライバ

Windowsプログラミング用語
スタック（Stack）　プログラムの自動変数と呼び出された関数へ渡された引数を保持する領域．サイズは変化しないがデータの量はプログラム実行中に変化する．

ディスクの場合はこのように設定すればよい(LIST 2)．PATHの設定，インクルードファイル，ライブラリファイルの指定がおもな仕事だ．MS-DOS, Windows, MS-C, Windows-SDK，これらはMS-DOS実行ファイルを持つので，PATHを設定することになる．インクルードファイル，ライブラリファイルはMS-C, Windows-SDKのそれぞれにあるので，両方設定してしまう．

DOSSHELL（注2）は入れないほうが身のためかもしれない．DOSSHELLがあると，ついメニューからコマンドプロンプトを選びたくなる．これが落とし穴で，あとでWindowsをデバッグモードにするときに環境変数領域が足りなくなって，身動きがとれなくなる．

あと，PS/55Zに限った注意ではあるが，WindowsSDKのPATHはDOSよりも前に設定する必要がある．PS/55ZにおいてSWITCH. EXEは日本語・英語のモード切り替えのプログラムだ．ところがWindowsSDKにも同名異機能のプログラムがあり，Windowsデバッグモード設定用のバッチファイルがこれを呼び出している．したがって，DOSのPATHが先にあると，デバッグモード設定のとき，

switch to japanese mode ?

などという間抜けな質問が現れて，あっけにとられることになる．

LIST 2

Windows 3.0開発用AUTOEXEC.BAT

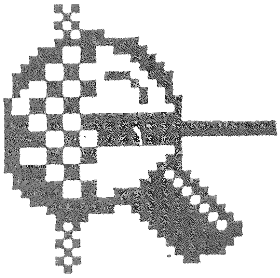
```
 1:  @echo off
 2:  set sys=c:
 3:  set comspec=c:¥ps2¥command.com
 4:  prompt $p$g
 5:  path c:¥windev;c:¥ps2;c:¥dos;c:¥tool;c:¥c600¥binb;c:¥c600¥bin;
 6:  proprint on
 7:  graphics graphics
 8:  set lib=c:¥c600¥lib;c:¥windev¥lib;
 9:  set include=c:¥c600¥include;c:¥windev¥include;
10:  set helpfiles=c:¥windev¥sdkadv.hlp
11:  set init=c:¥c600¥init
12:  mouse
```

2,3,4: DOS4.0用の設定
5: パスをDOS, Windows, MS-C 6.0, Windows開発キットのそれぞれに通す
8: MS-C 6.0とWindowsSDKのライブラリ設定
9: MS-C 6.0とWindowsSDKのインクルードファイル設定
10: PWB上で参照するヘルプファイルの設定
11: PWBが参照するPWB初期化ファイルのディレクトリ設定
12: PWBで使用するマウスドライバの起動

Windowsを組み込む
↓
MS-C 6.0を組み込む
↓
TOOLS.INIを作る
↓
Windows開発キットを組み込む
↓
CONFIG.SYSを設定する
↓
AUTOEXEC.BATを設定する

図1 インストールのダイアグラム

(注2) DOSSHELL　IBM DOS4.0Jのシェルプログラム．



ディレクトリツリー

最終的にできあがったディレクトリツリーを示す(図2). もし, ディレクトリの迷路にまよったら参照してほしい.

サンプルソースのコンパイル

これですべての作業が完了した. さっそくGENERIC. Cをコンパイルしてみよう. GENERIC. CはWindows-SDKについてくるサンプルソースで, ウインドウ作成だけを目的としている.

DOS上でのコンパイル

とりあえずはPWB上でもWindows上でもなく, DOSのコマンドラインからコンパイルすることにする. システムリセットののち, まずサンプルプログラムのあるディレクトリに移

図2
インストール後のディレクトリツリー

```
C:¥ ┬ DOS ──────────────────────── DOS4.0システムファイル
    ├ ETC
    ├ PS2
    ├ C600 ──┬ BINB
    │        ├ BIN
    │        ├ LIB ──────────────── MS-C6.0ライブラリファイル
    │        ├ INCLUDE ── SYS ───── MS-C6.0インクルードファイル
    │        ├ INIT ─────────────── PWB初期化ファイル
    │        ├ HELP ─────────────── PWB用MSC6.0ヘルプファイル
    │        └ SOURCE ┬ SAMPLES
    │                 └ DOC
    ├ WINDOWS ─ SYSTEM ──────────── Windows3.0 ファイル群
    ├ WINDEV ┬ LIB ──────────────── SDKライブラリファイル
    │        ├ DEBUG ────────────── デバッグバージョンファイル
    │        ├ NODEBUG ──────────── 通常バージョンファイル
    │        ├ INCLUDE ── SYS ───── SDKインクルードファイル
    │        └ SAMPLES ┬ BITMAP ──── ビットマップサンプル
    │                  ├ CLIPTEXT ── クリップボードサンプル
    │                  ├ CLOCK ───── 時計サンプル
    │                  ├ CURSOR ──── カーソルサンプル
    │                  ├ DDE ─────── DDEサンプル
    │                  ├ DEFPROCS ── デフォルト手続きサンプル
    │                  ├ EDITCNTL ── エディットコントロールサンプル
    │                  ├ EDITFILE ── ファイル操作サンプル
    │                  ├ EDITMENU ── メニュー操作サンプル
    │                  ├ FILEOPEN ── ファイルオープンサンプル
    │                  ├ GENERIC ─── GENERICサンプル
    │                  ├ HELPEX ──── ヘルプファイルサンプル
    │                  ├ ICON ────── アイコンサンプル
    │                  ├ INPUT ───── 入力サンプル
    │                  ├ MEMORY ──── メモリ管理サンプル
    │                  ├ MENU ────── メニューサンプル
    │                  ├ MULITIPAD ─ 簡易エディタサンプル
    │                  ├ MYPAL ───── パレットサンプル
    │                  ├ OUTPUT ──── ウィンドウ出力サンプル
    │                  ├ OWNCOMBO ── OWNコンボボックスサンプル
    │                  ├ PRNTFILE ── プリンターサンプル
    │                  ├ RAINBOW ─── カスタムコントロールサンプル
    │                  ├ SELECT ──── DLLサンプル
    │                  ├ SHOWDIB ─── DIB,パレットサンプル
    │                  ├ SHOWFONT ── フォント表示サンプル
    │                  └ TTY ─────── 通信サンプル
    └ TOOL ──────────────────────── 開発ツール類
```


Windowsプログラミング用語
スプーラ（Spooler） マルチタスクシステムにおいて複数のタスクからのプリント要求をスタックする機能. 同時にプリントデータのバッファリングも行う.

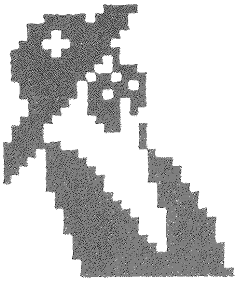
動する.

C>cd ¥windev¥sample¥generic

実際のコンパイルには, NMAKE.EXEを使う. NMAKEとは, MS-C 5.1までのMAKE.EXEのニューバージョンである. すでにAUTOEXEC.BATでPATHを設定してあるので, どこからでも呼び出せるはずだ. MAKEファイル名は拡張子なしのGENERICだから次のように入力すればコンパイルが始まる.

C> nmake generic

画面1のように表示されたのち, EXEファイルができあがる. さっそくWindowsを立ち上げよう. ファイルマネージャー上からGENERICを動かして完成だ. いったんコンパイルに成功したらしめたもの. あとはこの状態を発展させてゆくだけでいい.



PWB上でコンパイルしてみる

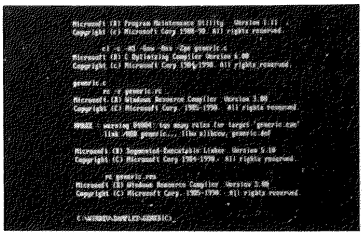
先に述べたように, MS-C 6.0にはPWBという統合環境が用意されている. ちょうどQuickCの統合環境でMS-Cが使えるような構造になっている. とはいっても, コンパイルやリンク機能がPWB自体に内蔵されているわけではない. PWBはコンパイラやリンカを, 簡単に呼び出す機能を持っているだけだ. これまでのMS-Cではコンパイルやリンク時の/オプションを使うにはマニュアルと首っ引きであったのが, PWB上ではダイアログボックス内のスイッチやチェックボックスの操作で簡単に設定できる.

PWB固有の機能としては, エディタ機能, ヘルプ機能があげられる. エディタはサブセット版ではなく, 必要な機能がすべてそろっている. ヘルプ機能もQuick Cなみの便利さで, SDKに付属するPWB用のヘルプファイルを使用すれば, Windows関数のヘルプも表示させることができる. また, 関数名上で[SHIFT]+[F1]キーを押せば, 関数リファレンスが表示されるなど芸が細かい.

それでは実際にPWB上でサンプルソースをコンパイルしてみよう. 今度もGENERICで挑戦する. GENERIC, GENERIC.C, GENERIC.H, GENERIC.RC, GENERIC.DEF以外のファイルが同じディレクトリにある場合は削除しておこう. とにかくコンパイルするという目標を達成するには, 次の3つの手続きが必要となる.

(1) Build OptionsとしてWindows実行可能形式を設定する

まず, コマンドライン上でPWB [return] と入力してPWBを起動する. エディット画面が表示されたら, "options"メニューの"Build Options"をえらび, "Set Initial Build Options..."で, "Windows EXE"を選択する. "Windows EXE"の他には, DOS上のEXEファイルやOS/2用のファイル群, WindowsのDLLなど, さまざまなプログラム作成環境を設定することができる. 今回は"Windows EXE"として, Windows用の実行可能ファイルを作成することになる.

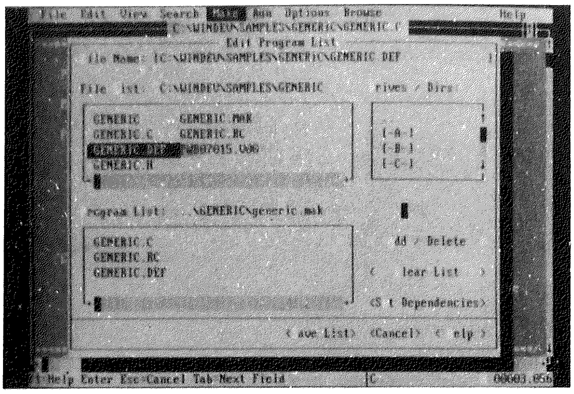


画面1　DOSコマンドラインからのコンパイル

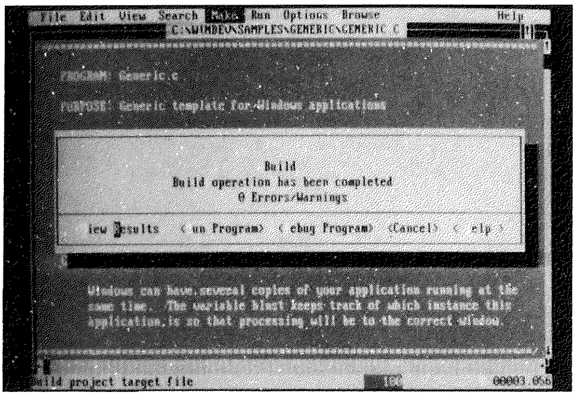
(2) MAKEファイルの自動作成をする

"Make"メニューから"Set Program List"を選択する. するとダイアログボックスが表示される. 最初に必要なのが, カレントディレクトリの移動で, "¥WINDEV¥SAMPLES¥GENERIC"に設定する. 次に, "File Name:[*.mak....]とあるので, "*"の部分を"GENERIC"と書き換え, "GENERIC.mak"とする. すると"File does not exist!"と警告され, "C:¥WINDEV¥SAMPLES¥GENERIC¥GENERIC.MAK Create? 〈Yes〉〈No〉〈Help〉"と聞いてくるので, 〈Yes〉と答える. するとファイルリストのなかに, カレントディレクトリに存在するファイル群が表示されるので, "GENERIC.C", "GENERIC.RC", "GENERIC.DEF"の3つを選択する. そして"〈Save List〉"をダブルクリックすれば, 自動的にMAKEファイルが作成される.

Windowsプログラミング用語
スペーシング(Spacing)　テキストをディスプレイに表示する行送り. Windowsはテキストもビットマップで表示するのでスペーシングサイズは固定ではない.



画面2　PWBの画面：MAKEファイル作成



画面3　PWBの画面：コンパイル

(3) MAKEファイルの内容にしたがってコンパイルする

"Make"メニューのなかから"Build"を選ぶ．するとコンパイルとリンクが始まる．"Build operation has been completed"と表示されたら，完成である．ダイアログボックス内の"<Run Programs> "を選ぶと，Windows 3.0がPWB上から立ち上がり，自動的にGENERICが実行されることとなる．

WindowsとPWB，どちらを先に起動したらよいか？

PWBを使うには２つの方法がある．

(1) PWBを起動した後，PWB上からWindowsを呼び出す方法

(2) Windowsを起動した後，Windows上からPWBを呼び出す方法

(2)のWindowsからPWBを呼び出す方法はさらに，スタンダードモードから呼び出すのとエンハンストモードから呼び出すのと２つの選択ができる．それぞれの方法には，一長一短があり，どれを選んだらよいか悩むところだ．

まず，PWBを先に起動することを考えてみよう．この方法の最大の長所は，実行を含むすべての動作をPWBのメニュー上から行えることである．PWBの統合環境のひとつとして，Windowsが入り込むわけだ．ただし，PWBからWindowsへの切り替えスピードはかなり遅い．毎回PWB上にある情報をディスクに退避させて，Windowsを最初から起動するからだ．

Windowsを先に起動する方法では，ファイルマネージャかプログラムマネージャからPWBを呼び出すことになる．PWBでコンパイルした後は，タスクスイッチングをしてWindowsに戻る．PWBを先に起動した場合と比較すると，スイッチ動作が速いことが長所だ．ただし，PWBの統合環境は完全には生かされない．

また，この方法をとった場合スタンダードモードとエンハンストモードでは，速度が微妙に異なってくる．スタンダードモードではPWB呼び出し時にWindowsで使用しているメモリをディスクへスワップするため多少の時間がかかる．この点，エンハンストモードではPWBが仮想8086モードで動作するので，メモリに余裕さえあれば切替えは一瞬である．ただし，スタンダードモードよりタスク管理に多少時間がとられてしまう．その結果コンパイルや実行スピードが少々犠牲になる．

私自身の好みは，Windowsをエンハンストモードにしてそこから PWBを呼び出す方法だ．一瞬で終るタスクスイッチングはやはり魅力的である．

CHART

PWBとWindowsの関係

先に起動するもの	PWB	Windows スタンダードモード	Windows エンハンストモード
PWBの統合環境	有効利用	不完全	不完全
切替えスピード	かなり遅い	遅い	速い
コンパイル／実行スピード	普通	普通	やや遅い

Windowsプログラミング用語
スラッシング（Thrashing）　マルチタスクで他のタスクのコードをディスクなどからロードする時間をスラッシングという．

デバッグについて

デバッグバージョンのインストール

Windowsにはデバッグバージョンとノーマルバージョンの2種類がある．デバッグ時にはWindowsカーネルをSDKに付属するデバッグバージョンに置き換える．こうするとエラー発生時にはシリアルポートにエラーコードが送られる．この機能を利用するにはシリアル端末が必要となる．

デバッグバージョンとノーマルバージョンの切り替えは，Windows2.11のときと比べると簡単になっている．N2D.BATとD2N.BATという2つのバッチファイルが用意されているので，迷うことはないだろう．N2Dはノーマルモードからデバッグバージョンへの切り替えに，D2Nはその逆に使用する．

私はここで恐ろしく時間をくってしまった．このバッチファイルは中でSWITCH.BATという別のバッチファイルを呼び出しているのだが，これがPS/55ZのSWITCHコマンドとまっこうからぶつかってしまうのである．これを解決するにはDOSのPATH設定をSDKよりも後にすればよい．また，あまりにも間の抜けた話だが，PC-DOS4.0に付属するDOS-SHELLのコマンドモードメニューからN2D.BATを起動していたため，環境変数領域が足りなくなるという間違いも侵していた．「AUTOEXEC.BATを作る」のところでDOS-SHELLを除いておいたほうが賢明だといったのはそういう意味なのである．

デバッガの進化

Windows 3.0ではデバッグ環境が大いに進化している．Windows用のコードビューであるCVWがフルスクリーンで使用できるようになったのだ．これまではせっかくのCVWもコマンドモードでしか使用できなかったため，使い勝手が悪かった．Windows 3.0用のCVWはDOS上のCVと同様，ウィンドウ表示のデバッグ環境として使用することができる．たとえば，CVW上に表示されているソースコードをダブルクリックすることにより，ブレークポイントを設定できる．また，メッセージにブレークポイントを設定する機能も，便利に利用できる．変数内容の表示も，これまた簡単だ．

さて，現在のWindows 3.0でCVWを使用するためには，白黒ディスプレイアダプタあるいは8514/Aディスプレイアダプタが必要となる．日本語版のSDKでは画面切り替えによるCVWがサポートされるそうなので，より身近にCVWが利用できるだろう．

CVWの起動は簡単，プログラムマネージャーに登録された虫眼鏡風のコードビューアイコンをクリックするだけだ．あとは2つのモニタ間をいったり来たりするのだが，両方同時には操作できない．Breakキーを押すかまたはブレークポイントにさしかかると，制御がCVWにうつる．CVWにWindows続行コマンドを送ると再びWindowsに制御がうつることになる．

Windows 3.0のデバッガには，このほかSYMDEBとWDEB386が存在する．リアルモードのデバッグにはSYMDEBを，80386アーキテクチャ固有のデバッグにはWDEB386を使用することになる．

実用的な開発環境

はっきりいって，今回の環境はインストールをやってみせるだけのもので，本格的なWindowsアプリケーション開発には役不足だ．PS/55Zというマシン自体はいいのだが，ハードディスクはせめて60MBタイプのものが必要だし，メモリも4～6MBはほしい．RAMディスクとキャッシュディスクをうまく使い分けて，できるだけハードディスクのアクセスを減らせば，今回の環境に比べて数倍の効率が期待できるだろう．ソースファイルはハードディスク，テンポラリファイルはRAMディスク，さらにディスクキャッシュを使うのがベストだと思う．

Windows開発環境の構築には，多少の経験と知恵が必要だが，今回の記事が，それに多少なりとも貢献できれば幸いだ．

今回参考としたマニュアル：
Windows:User's Guide
MS-C:Installing and Using the Professsional Development System
MS-C:Reference
MS-C:Advanced Programming Techniques
SDK:Installation and Update Guide
SDK:Reference-Volume2
SDK:Tools
SDK:Guide to Programming
SDK:IBM Systems Application Architecture Common User Access Advanced Interface Design Guide

Windowsプログラミング用語
ダイアログボックス（Dialog boxes）　チャイルドウィンドウの一種で，おもにユーザーからの入力を受け付けるのに使われる．

Microsoft Windows Device Development Kit (DDK)

西川和久

Windows Device Development Kit (DDK)は，MS-Windows用のデバイスドライバを書くときに必ず必要になるものである．今回Windows3.0用DDKを入手したので，簡単に紹介したい．

ハードディスクは20MB以上必要！

私の手元に届いたのは5インチ版(1.2MB)で，枚数はなんと10枚にもおよび各ファイルに圧縮がかかっているので，ハードディスクへ復元すると20MB以上の容量をとってしまう．Windows用プログラムの開発には，DDK以外にMicrosoft C，SDK，Windows3.0などが必要になるので，通常自分が使っている開発ツールを合わせると，軽く100MBを越えてしまう．ハードウェアは，CPUパワーも含め，かなりのものを準備する必要がある．

私は運よく（?）クボタコンプスのTAHITIというMO(5インチ片面500MB，両面で1GB，平均シークタイム30msの高速MO(光磁気ディスク)で，DiskCacheと併用するとメインのディスクとして使える優れものである)を買っていたので，容量を気にせずインストールすることができたが，普通のプログラマがこれだけのハードウェアを揃えるのはかなり難しいと思われる．ディスクの内容については，図1を参考にしていただきたい．

内容は，オリジナルのビデオ／プリンタ／キーボード／RS-232C／マウス／フォント／サウンドなど，Windows専用ドライバを書くために必要なドキュメント，サンプルソースコード，開発ツール，ドライバテストツールであり，ディスクのうちの2枚にはVirtual Device，すなわちDPMI関係のものが収められている．

ディスプレイドライバ関係

ディレクトリを見ると気がつくのが，1，4，8という3つのプレーンにわかれてファイルが収められているということだ．この3つのプレーンというのは，モノクロ，16色，256色対応のことだと思えばよい．一般的なEGAやVGAは16色，すなわち4プレーンの下にドライバのソースコードがある．また，Video SevenのSVGAでは256色の対応をしているので8プレーンの下にある．

このようにWindows 3.0が標準でサポートしているすべてのディスプレイドライバが完全なソースコードで入っているため，PCに対応しているビデオボードの移植はかなり簡単（?）にできそうである．

プリンタドライバ関係

ディスプレイドライバ同様，Windows 3.0が標準サポートしているプリンタドライバの全ソースファイルが収められている．

それぞれ，HPPCL系，PostScript系，Raster系（ESC/P，ESC/P，Color）などがある．

キーボード／マウス／RS-232C／フォント／サウンド関係

5枚目から7枚目にかけて3枚のディスクに収められており，ネットワーク関係のドライバ，インクルードファイル，ライブラリなどもここに含まれる．

開発サポートツール／ドライバテストツール

ドライバテストツールは7枚目のディスクに収められている．また，サポートツールには図2のものがあるが，時間の関係上実際に使用していないので，ディレクトリの内容だけを示しておく（ただし，ファイルサイズは圧縮したままのものである）．

ファイルの数は2000以上に及ぶ

時間の関係上簡単な内容紹介だけになってしまったが，とにかくひと通り眺めるだけでも，かなり時間がかかってしまう（ファイル数は2000以上にも

Windowsプログラミング用語
ダイナミックデータ変換（DDE）　タスク間で動的にデータを交換するプロセス間通信を実現するプロトコルの一種

及ぶ).

Windows用のアプリケーションを書くだけでもたいへんなのに，これがドライバになると，もはや嫌気がさして作る気力が失せてしまうというのが本音ではないだろうか！？ましてや，国産のマシンのようにIBM PCとまったく互換性がないパソコンではなおさらである…….

■編集部より

Windows 2.XまでのDDKは，デバイスドライバ開発契約を結んだソフトハウスなどに供給されていたが，Windows 3.0からは一般に販売することになった.日本におけるDDKの供給も米国にならって，Windows 3.0からは商品として販売されるようである．英語版はマイクロソフト（株）を通して入手可能であるが，日本語版については供給の方法，価格などが現在検討されている．

■Device Driver Samples & Tools Disk 1
ボリューム WIN30DDK01 のディレクトリパスリスト

```
A:.
+---DISPLAY
    +---1PLANE
    |   +---BITBLT
    |   +---BW
    |   |   +---CGAHERC
    |   +---RC-HIGH
    +---4PLANE
        +---BITBLT
        +---COLOR
        |   +---EGA
        |       +---EGAHIRES
        |       +---VGA
        +---RC-HIGH
        +---RC-MED
```

■Device Driver Samples & Tools Disk 2
ボリューム WIN30DDK02 のディレクトリパスリスト

```
A:.
+---DISPLAY
    +---1PLANE
    |   +---BLTINCS
    |   +---BW
    |   |   +---CGAHERC
    |   |   |   +---CGA
    |   |   |   +---COMPAQ
    |   |   |   +---HERC
    |   |   +---EGA
    |   |       +---EGAHIBW
    |   |       +---EGAMONO
    |   |       +---MCGA
    |   +---LOGO
    |   +---RC-LOW
    |   +---RC-MED
    +---8PLANE
        +---V7
            +---RES
                +---96X96
```

■Device Driver Samples & Tools Disk 3
ボリューム WIN30DDK03 のディレクトリパスリスト

```
A:.
+---PRINTERS
|   +---HPPCL
|       +---FNT
|       +---INK
|       +---RC
|           +---INTL
|           +---PCM
+---DISPLAY
    +---8PLANE
        +---V7
        +---8514
            +---RESOURCE
            +---SOURCE
```

■Device Driver Samples & Tools Disk 4
ボリューム WIN30DDK04 のディレクトリパスリスト

```
A:.
+---PRINTERS
    +---FINSTALL
    |   +---INK
    |   +---RC
    |   +---SRC
    +---HPPCL
    |   +---RC
    |   +---SRC
    +---PSCRIPT
        +---RES
```

■Device Driver Samples & Tools Disk 5
ボリューム WIN30DDK05 のディレクトリパスリスト

```
A:.
+---KEYBOARD
|   +---TABLES
|       +---MT
+---NET
+---PRINTERS
    +---RASTER
    |   +---COLOR
    |   |   +---IBMCOLOR
    |   +---EPSON
    +---PSCRIPT
        +---RES
        +---RESOURCE
        +---UTILS
```

■Device Driver Samples & Tools Disk 6
ボリューム WIN30DDK06 のディレクトリパスリスト

```
A:.
+---COMM
+---INCLUDE
+---KEYBOARD
+---LIB
|   +---C51
+---MOUSE
+---SOUND
+---TESTS
    +---DTA
    +---PRINT
        +---BASIC
```

■Device Driver Samples & Tools Disk 7
ボリューム WIN30DDK07 のディレクトリパスリスト

```
A:.
+---TESTS
|   +---HCT
|   +---ITE
|   |   +---BIN
|   |   +---DISPTEST
|   |   +---DOCS
|   |   +---ISG_TEST
|   |   +---PRNTTEST
|   |       +---FRAME
|   +---NETTEST
|   +---PRINT
|       +---BASIC
|       +---FONTS
|       +---QAD
+---FONTS
    +---FIXED
    |   +---8514
    |   +---CGA
    |   +---EGA
    |   +---VGA
    +---SYSTEM
        +---8514
        +---CGA
```

■Device Driver Samples & Tools Disk 8
ボリューム WIN30DDK08 のディレクトリパスリスト

```
A:.
+---FONTS
|   +---SYSTEM
|       +---CGA
|       +---EGA
|       +---VGA
+---GRABBERS
|   +---CGAHERC
|   |   +---CGA
|   |   +---HERCULES
|   +---EGA
|   |   +---EGACOLOR
|   |   +---EGAMONO
|   +---VGA
|       +---VGACOLOR
|       +---VGAMONO
+---SETUP
+---TOOLS
```

■Virtual Device Samples & Tools Disk 1
ボリューム WIN30VDDK01 のディレクトリパスリスト

```
A:.
+---EBIOS
+---INCLUDE
+---PAGESWAP
+---TOOLS
|   +---DEBUG
|   +---NODEBUG
+---VDDCGA
+---VDDEGA
+---VDDHERC
+---VDMAD
```

■Virtual Device Samples & Tools Disk 2
ボリューム WIN30VDDK02 のディレクトリパスリスト

```
A:.
+---BIOSXLAT
+---COMBUFF
+---GRABBERS
+---LDOSNET
+---OEMFONTS
+---VCD
+---VFD
+---VHD
+---VKD
+---VMD
+---VNETBIOS
+---VPD
```

図1 Microsoft Windows Device Development Kit のディレクトリパスリスト

```
 ドライブ F のボリュームは WIN30DDK08  です
 ディレクトリは  F:¥TOOLS

AFM      EXE    16508 90-06-01    3:00
CMPCTFON BAT      155 90-06-01    3:00
LINK4    EXE    68539 90-06-01    3:00
MKPRN    EXE    24421 90-06-01    3:00
PFMEDIT  EXE    22480 90-06-01    3:00
SMASH    EXE     9178 90-05-01    3:00

 ドライブ F のボリュームは WIN30VDDK01 です
 ディレクトリは  F:¥TOOLS

ADDHDR   EXE     5709 90-06-01    3:00
LINK386  EXE    82804 90-06-01    3:00
MAPSYM32 EXE    23323 90-06-01    3:00
MASM5    EXE    81389 90-06-01    3:00
WDEB386  EXE    46624 90-06-01    3:00
```

図2 サポートツール

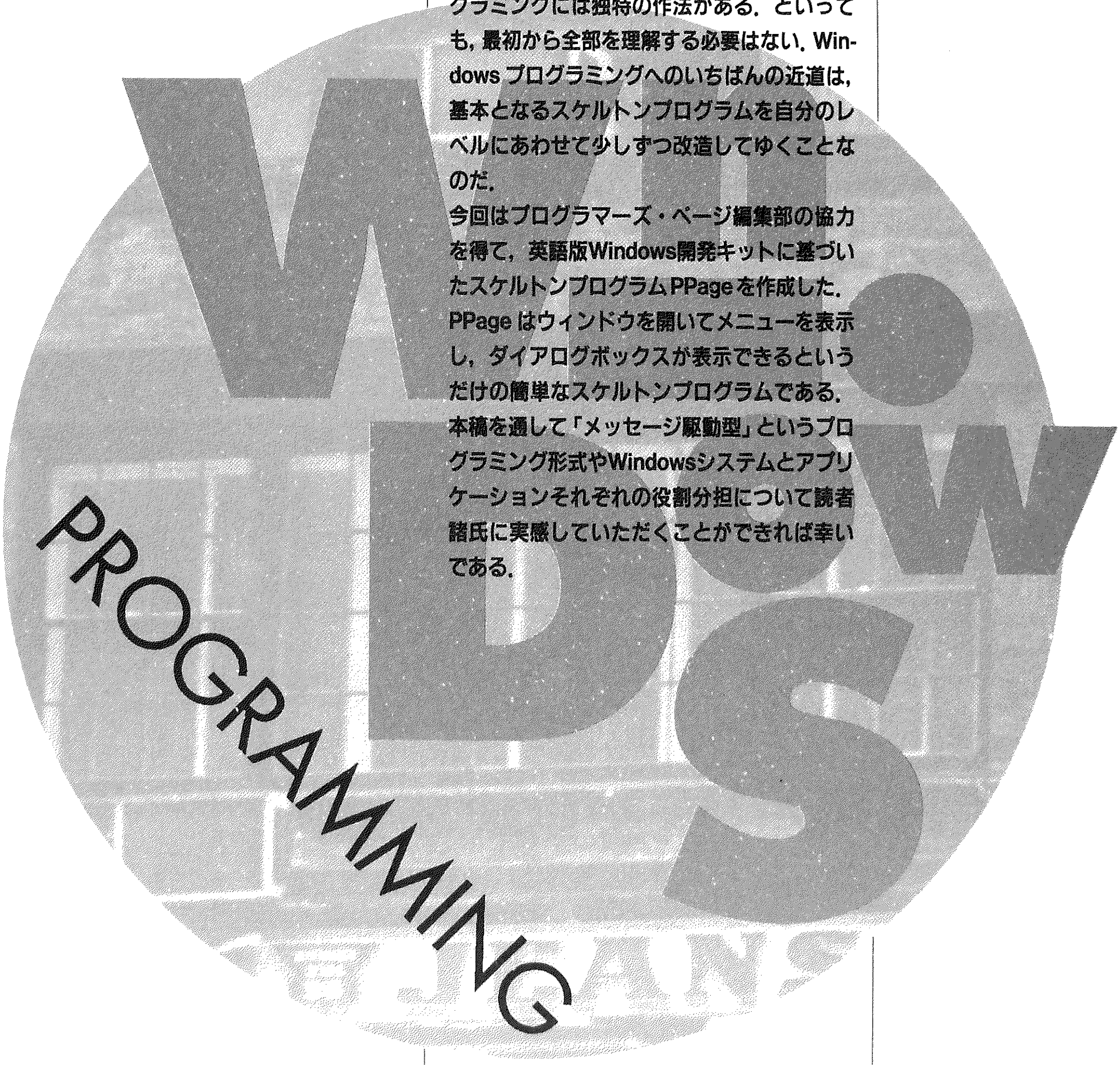
Windows
プログラミング講座
3

Windows 基本プログラム大解剖

永井知己

「Windows プログラミングの基本はスケルトンから」。Windowsのプログラミングに慣れた人が，口をそろえていう言葉である．DOS 上でのプログラミングと違って，Windowsプログラミングには独特の作法がある．といっても，最初から全部を理解する必要はない．Windows プログラミングへのいちばんの近道は，基本となるスケルトンプログラムを自分のレベルにあわせて少しずつ改造してゆくことなのだ．

今回はプログラマーズ・ページ編集部の協力を得て，英語版Windows開発キットに基づいたスケルトンプログラムPPage を作成した．PPage はウィンドウを開いてメニューを表示し，ダイアログボックスが表示できるというだけの簡単なスケルトンプログラムである．本稿を通して「メッセージ駆動型」というプログラミング形式やWindowsシステムとアプリケーションそれぞれの役割分担について読者諸氏に実感していただくことができれば幸いである．



1 WindowsアプリケーションPPage 概要解説編

WindowsアプリケーションPPageの概要

WindowsアプリケーションPPageは，Windowsアプリケーションが必ず使用するごく基本的な機能だけを盛り込んだプログラムである．PPageの仕様はたったこれだけだ．

(1)　PPageウィンドウを開く
(2)　Helpというメニューを持つ
(3)　Helpが選択されるとAbout PPageというポップアップメニューを表示する
(4)　About PPageメニューが選択されるとAbout PPageダイアログボックスを表示する

順序は逆になるがイメージを理解してもらうために，先にPPageを実行した画面を見てもらうことにしよう．

PPageプログラムで使用するファイル

PPageというWindowsアプリケーションのために次の5つのファイルを作成した．ヘッダファイルppage.h，Cソースプログラムファイルppage.c，メイクファイルppageについては，DOS上でCプログラムの場合と基本的な役割は同じだ．Windowsアプリケーションではこれに加えて.rc，.defという拡張子をもったファイルが必要となる．これらのファイルはそれぞれリソース定義ファイル，モジュール定義ファイルと呼ばれる．

リソース記述ファイルにはアプリケーションで使用するアイコン，カーソル，メニュー，ダイアログボックスなど各種リソールの定義を記述する．ppage.rcにはPPageプログラムで使用するメニューとダイアログボックスに関して記述されている．モジュール定義ファイルにはアプリケ

CHART

PPageプログラムの作成に必要なファイル

ファイル名	内容
ppage.h	PPageヘッダファイル
ppage.c	PPageソースプログラム
ppage.rc	PPageのリソース記述ファイル
ppage.def	PPageのモジュール定義ファイル
ppage	PPageのメイクファイル

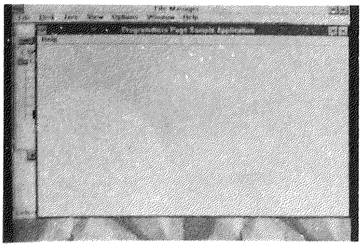


写真1　PPage実行

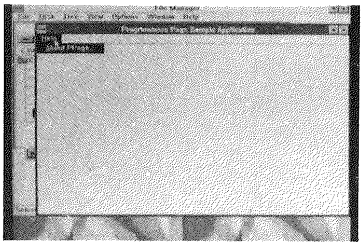


写真2　PPageのHELPメニューを選択したところ

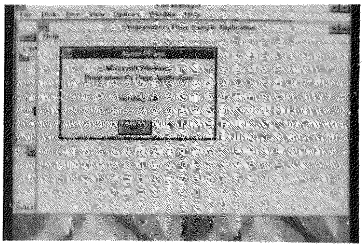


写真3　About PPageダイアログボックス表示

Windowsプログラミング用語
ダイナミックリンクライブラリ（DLL）　Windowsの実行ファイルはメモリにロードされてからもリンクを行う．DLLはリンクされる共有ルーチンのこと．

ーション名，セグメント，必要なメモリ，プログラムの外から呼ばれる（エクスポートされる）ルーチン名などを記述する．ppage. defではWndProc()，AboutProc()という2つの関数名がこれにあたる．

ソースプログラム ppage.cの構造

本題の大解剖に移る前に，まずppage. cの構造にふれておこう．本稿で紹介するppage. cは大きく3つの部分に分けられる．第1の部分はDOS-Cのmain() 関数にあたるWinMain() 関数である．本文の中ではWinMain() 関数本体と，そこから呼ばれる2つの関数InitApp()，InitInst()に分けてリストを掲載している．第2の部分はウィンドウ手続きである．これはWndProcという関数名になっている．第3の部分はPPageプログラムで表示するダイアログボックスに関する処理を行う関数AboutProcである．

Windowsアプリケーションの構造は，一般のDOS-Cプログラムと若干の違いがある．DOS-Cプログラムでは，まずmain() 関数に制御が移る．すべての関数はプログラム中の別の関数から呼ばれていて，元をたどれば必ずmain()関数にたどりつく．ところがWindowsアプリケーションでは，一見どこからも呼ばれていないように見える関数が存在する．今回のppage. cでもウィンドウ手続きWndProc()，ダイアログボックス手続きAboutProc()を直接呼び出す命令はない．これらの関数はWndProc () からではなく，Windowsシステムから直接呼ばれるのである．

PPageの実行

次に，実際にこれらの関数がどのような順序で実行されていくのかを説明しよう．PPageを実行すると，制御はまずWinMain() に移る．WinMain() ではまずPPageアプリケーション用のウィンドウを開くためのさまざまな初期化処理を行う．そのおもなものがInitApp() 関数，InitInst() 関数である．これらの関数は通常のDOS-Cの場合と同様，WinMain() から呼ばれる．

初期化処理が終ると，WinMain() はメッセージ待ちの状態に入る．Windowsアプリケーションは，DOSアプリケーションのように単独では動けない．常にWindowsシステムからメッセージを受け取りそれを処理する．「ウィンドウを閉じなさい」というメッセージがくるまでこの処理は繰り返されるので，メッセージループと呼ばれる．

Windowsから受け取ったメッセージを実際に処理するのはウィンドウ手続きWndProc() の役割である．WinMain() からWndProc() へどのようにして制御が移るのかは後半の大解剖編で詳しく説明するのでここでは省略する．WndProc() 内部ではメッセージの内容によって処理が分岐する．「About PPageメニューが選択されました」というメッセージがきたときは，ダイアログボックス手続きを呼び出す準備をしてWindowsシステムに制御をもどす．それ以外のメッセージの処理についてはWindowsシステムにまかせてしまう．

ダイアログボックス内部での処理はダイアログボックス手続きAboutProc() で行うのだが，AboutProc() はWndProc() から呼ばれるのではなく，WndProc() がWindowsシステムに制御をもどした後，Windowsシステムから呼び出される．

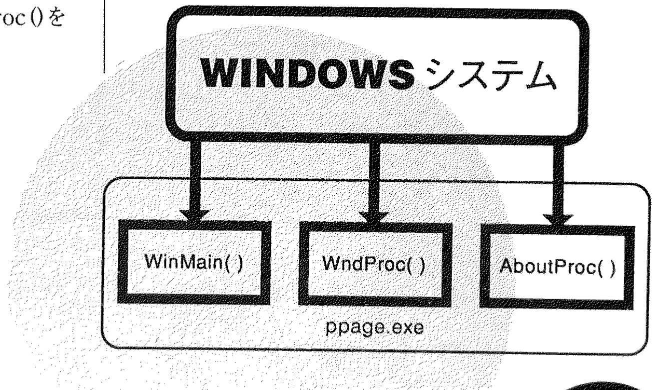
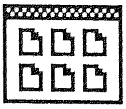


図1 ppageプログラムではWindowsシステムから3つの関数が呼び出される

Windowsプログラミング用語
チェックボックス（Check box）　ユーザーがオプションの選択をする場合などに使われる．チェックボックスのボタンはトグルスイッチになっている．

2 WindowsアプリケーションPPage 大解剖編

ここでは，Windowsの基本プログラムPPageを例に，そのメインプログラムであるppage.cで使っているWindows関数を中心に，その働きを図で示しながら解説します．ソースと解説を対応させるためにppage.cは4つのブロックに区切っています．また，ヘッダーファイルやリソースファイルなどppage.c以外のファイルについても解説しています．



ppage.c ・・・・・・・・・・・・・・・・・1

```
/**************************************************************************
    Programmer's Page sample source program
**************************************************************************/

#include "windows.h"            /* どんなアプリケーションにも必要          */
#include "PPage.h"              /* このプログラム用のヘッダファイル        */

HANDLE hInst;                   /* 現在動作中のインスタンス                */

/**************************************************************************
    MAIN PROCEDURE
**************************************************************************/

int PASCAL WinMain(hInstance, hPrevInstance, lpCmdLine, nCmdShow)
HANDLE hInstance;               /* 現在のインスタンス                      */
HANDLE hPrevInstance;           /* 一つ前に実行されたインスタンスハンドル  */
LPSTR lpCmdLine;                /* コマンドライン文字列へのロングポインタ  */
int nCmdShow;                   /* アイコン指定で起動されたか否か          */
{
    MSG msg;                    /* メッセージ変数                          */

    if (!hPrevInstance)               /* これが最初のインスタンスか？      */
        if (!InitApp(hInstance))      /* すべてのインスタンス共通の初期化  */
            return (FALSE);           /* もし初期化に失敗した場合の脱出    */

/* それぞれのインスタンス固有の初期化を行う                                */
    if (!InitInst(hInstance, nCmdShow))
        return (FALSE);

/* WM_QUITメッセージが送られてくるまで，メッセージを受取り，分配する       */
    while (GetMessage(&msg,     /* メッセージ構造体                        */
            NULL,               /* メッセージを受け取るウインドウハンドル  */
            NULL,               /* 処理する最下位のメッセージ              */
            NULL))              /* 処理する最上位のメッセージ              */
        {
        TranslateMessage(&msg); /* 仮想キーコードを翻訳する                */
        DispatchMessage(&msg);  /* ウインドウへメッセージを分配する        */
    }
    return (msg.wParam);        /* PostQuitMessage()からの値を返して終了   */
}
```

●
●
●
●

windows.h

Windows.hは，ソフトウェア開発キットにより提供されるWindows専用のヘッダファイルでコンパイルの際には必ず必要となる．ここでは変数の型設定，#define，Windows関数の定義などWindows指定の定義のすべてが含まれている．たとえば，WM_DESTROYは次のように設定されている．

#define WM_DESTROY 0x0002

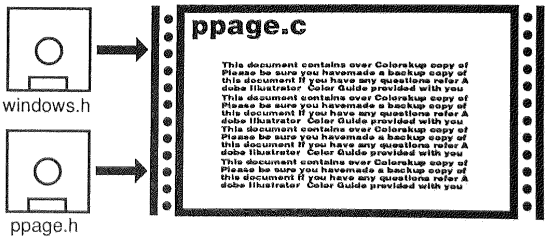
ppage.h

```
#define IDM_ABOUT 100

int PASCAL WinMain(HANDLE, HANDLE, LPSTR, int);
BOOL InitApp(HANDLE);
BOOL InitInst(HANDLE, int);
long FAR PASCAL WndProc(HWND, unsigned, WORD, LONG);
BOOL FAR PASCAL AboutProc(HWND, unsigned, WORD, LONG);
```

●
●

ppage.h

本プログラムのPPage用のヘッダファイル．#defineのほか，関数の定義などが含まれる．ppage.hでは#defineでIDM_ABOUTというメニューで使用する値を定義している．Windowsではアプリケーションの利用者があるメニューを選択すると，選択したメニューに応じた値がアプリケーションプログラムに渡されるというしくみになっている．PPageでは"About PPage..."というメニューを選択すると，WndProc()にユーザーがメニューを選択したことを示すWM_COMMANDメッセージとともにパラメータとしてIDM_ABOUTが送られる．



Windowsプログラミング用語
デバイスコンテキスト（Device context）　GDIによって管理されるデータ構造体で，プリンタやディスプレイ（内の特定のウィンドウ）と結び付けられている．

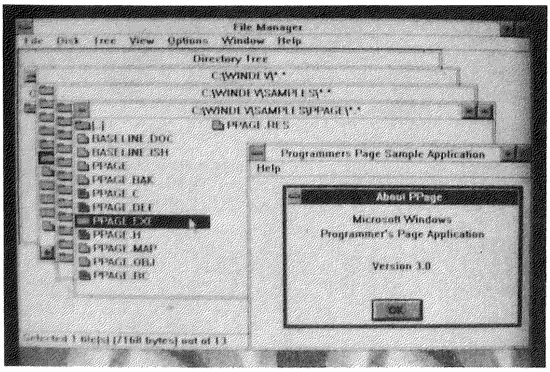
WinMain()

WinMain() はDOS-Cのmain() にかわるものである．アプリケーションがロードされるとまずここからスタートする．したがって，プログラマがWinMain() 以外の名前を勝手につけることはできない．また，WinMain() はPASCAL型の関数である．通常のCコール規約は呼び出す側にスタックポインタ調整のコードを生成するが，Pascalコール規約では呼び出される側にこのコードを生成する．したがって，呼び出す場所が多いほどPascalコール規約のほうがメモリ使用効率が向上する．

hInstanceというのは，インスタンスのハンドル，つまりWindowsシステムによってつけられたアプリケーションの識別番号だ．Windowsでは同じアプリケーションを複数起動することができる．たとえば，NotePadを3つ起動して異なるテキストファイルを編集できる．このときもNotePadのコードセグメントは1つしかメモリ上に存在しない．コードセグメントを共有して，データセグメントを3つ用意するという手法をとっているからだ．この場合のNotePadのコピーそれぞれをインスタンスと呼ぶ．先ほどのnotepadの例では3つのインスタンスが存在する．

hInstanceが自分自身のインスタンスのハンドルであるのに対し，hPrevInstanceは前に起動されたインスタンスのハンドルを意味する．lpCmdLineはコマンドライン文字列へのロングポインタである．これとWin. iniを利用して，*. TXTファイルをクリックするとNOTEPADが起動して同時にファイルがロードされるというようなオブジェクト指向環境を実現している．

CmdShowはアイコン指定で起動されたか否かの情報であり，後のShowWindow() とともに利用する．



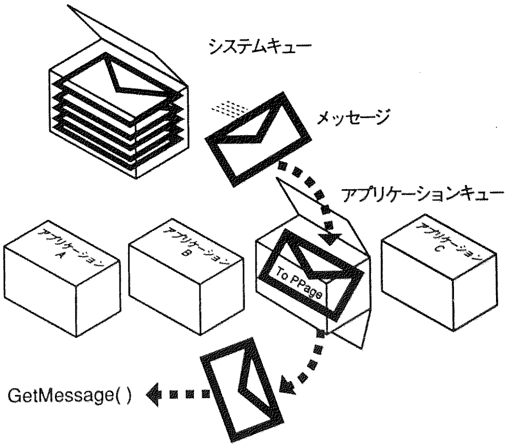
ファイルマネージャからPPageを起動したところ

WinMain() の実際の仕事は，最初のインスタンスだったばあい，InitApp() を呼び出すことと，すべてのインスタンスにつきInitInst() を呼び出し，ウィンドウ作成をすることである．また，これらの初期化がすんだ後メッセージループによりメッセージ処理を行う．

GetMessage()

Windowsはメッセージキューシステムを採用している．システムが生成したメッセージは，システムキューを通ったあとアプリケーションごとに存在するアプリケーションキューに分配される．メイン手続きはアプリケーションキューの前で，今か今かとメッセージが来るのを待っている．これがGetMessage() である．一度GetMessage() を実行すると次にメッセージが来るまでシステムに制御が移される．したがって，メッセージが到着するまでアプリケーションは次の動作を行えない．かわりにその間はほかのアプリケーションが自由に動作できる．

WinMain() がメッセージを受け取るとGetMessage() は値を返す．その値が真なら次の命令へ動作がうつり，偽ならアプリケーション終了のためにメッセージループから抜け出すことになる．実際にはWM_QUITメッセージが送られてきたときだけGetMessage() は偽の値を返す．したがって，通常のメッセージではTranslateMessage() 関数が次に実行される．

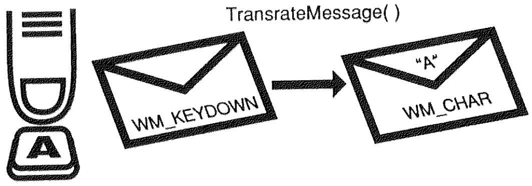


TranslateMessage()

TranslateMessage() とはキーが押された情報をメッセージに翻訳する関数だ．たとえば，キーが押されたという意味のWM_KEYDOWNメッセージを，キャラクタコ

Windowsプログラミング用語
バイナリラスタオペレーション（Binary raster operation）　ラスタオペレーションの中でも，とくに2つのピクセルパターンしかない論理操作のこと．

ードが入力されという意味のWM_CHARメッセージに翻訳したり，[F1]キーがおされたら，マウスでヘルプメニューを選んだのとおなじメッセージに翻訳したりする．この関数を使用することで，マウスがなくてもキーボードで操作できる環境を実現している．



DispatchMessage()

DispatchMessage() 関数によりメッセージはウィンドウ手続きに分配される．ウィンドウ手続きはメッセージを受け取りしだい，すぐにその内容を解釈して実行することとなる．実際にはswitch()文でこれを行うので構造自体は簡単である．

さて，DispatchMessage() がWndProcにメッセージを分配するのだが，ソースリストのどこ見回しても，WndProcをコールする場所がない．それはWndProcが今までのDOS上のCと異なり，Windowsシステムから直接呼ばれる形式だからだ．これをコールバック形式と呼ぶ．

DispatchMessage() 関数がWinMain() の中で実行されると，msg構造体の中で指定されたウィンドウをメッセージを引数としてWindowsシステムが呼び出すしくみとなっている．WndProc() がWindowsシステムから呼ばれるためには，あらかじめシステムにその場所を教えておかなければならない．クラス登録のときにWndProc() を登録し，またモジュール定義ファイル（ppage.def）の中のEXPORTSのなかに記載することにより，システムは確実にウィンドウ手続きを呼び出す．

以下にWndProc() が呼び出される手順を示す．

WinMain() の中のDispatchMessage() がシステムにWndProc() を呼び出すことを依頼

↓

システムはクラス登録された位置情報をもとにWndProc() を呼び出す

↓

WndProc() がシステムから呼び出される

↓

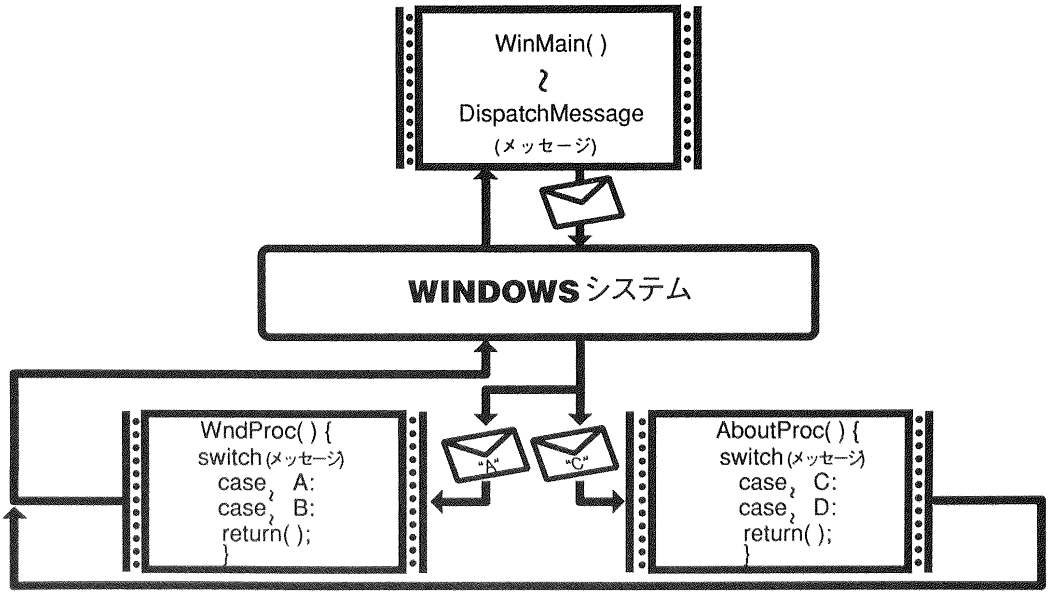
WndProc() の処理が終るとシステムにリターンする

↓

システムが実行をWinMain() に再びもどす

return(msg.wParam)

アプリケーションはWinMain() をreturnすることによって終了する．



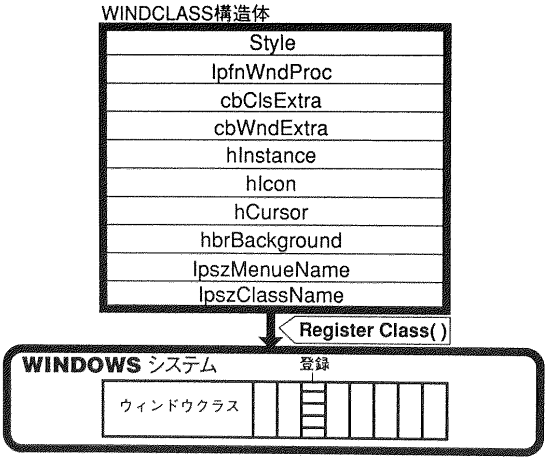
Windowsプログラミング用語
パスカルパラメータ引渡し規約（PASCAL parameter convention）　パラメータが呼び出される順にプッシュされ，関数自身がスタックの調整を行う規約．

ppage.c ・・・・・・・・・・・・・・・・・2

```
/*****************************************************************
    Initialize Apprication
*****************************************************************/

BOOL InitApp(hInstance)                      /* 現在のインスタンス           */
HANDLE hInstance;
{
    WNDCLASS  wc;

/* ウインドウクラス構造体をメインウインドウについての情報で満たす       */
    wc.style = NULL;                     /* クラスのスタイル             */
    wc.lpfnWndProc = WndProc;            /* メッセージを配るあて先       */
    wc.cbClsExtra = 0;                   /* クラス構造体の拡張バイト数   */
    wc.cbWndExtra = 0;                   /* 各ウインドウの拡張バイト数   */
    wc.hInstance = hInstance;            /* インスタンスハンドル         */
    wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground = GetStockObject(WHITE_BRUSH);
    wc.lpszMenuName =  "PPageMenu";   /* .RCファイル中のメニューリソース名 */
    wc.lpszClassName = "PPageWClass"; /* CreateWindow()で使うウインドウ名  */

/* ウインドウクラスを登録し，成功・不成功のコードを返す                 */
    return (RegisterClass(&wc));
}
/*****************************************************************
    Initialize Instance
*****************************************************************/

BOOL InitInst(hInstance, nCmdShow)
    HANDLE          hInstance;    /* 現在のインスタンスハンドル          */
    int             nCmdShow;     /* ShowWindow()の引数として与える      */
{
    HWND            hWnd;          /* メインウインドウのハンドル           */

/* インスタンスハンドルをstatic変数に保存する  この変数は，アプリケー      */
/* ションの中で何度も呼び出されることとなる                                 */
    hInst = hInstance;

/* 現在のインスタンスにおけるメインウイドウを作成する                       */
    hWnd = CreateWindow(
        "PPageWClass",             /* RegisterClass()で登録した名前         */
        "Programmers Page Sample Application",
                                   /* タイトルバーで表示される文字列        */
        WS_OVERLAPPEDWINDOW,       /* ウインドウの形                        */
        CW_USEDEFAULT,             /* ウインドウ左上のX座標                 */
        CW_USEDEFAULT,             /* ウインドウ左上のY座標                 */
        CW_USEDEFAULT,             /* ウインドウの幅                        */
        CW_USEDEFAULT,             /* ウインドウの高さ                      */
        NULL,                      /* 親ウインドウはありません              */
        NULL,                      /* メニューのハンドル                    */
        hInstance,                 /* インスタンスのハンドル                */
        NULL                       /* 付属するパラメータ                    */
    );

/* ウインドウ作成に失敗したらFALSEを返す                                    */
    if (!hWnd)
        return (FALSE);

/* ウイドウを表示 ; クライアントエリアを更新 ; 成功(TRUE)を返す             */
    ShowWindow(hWnd, nCmdShow);  /* ウインドウを表示する                    */
    UpdateWindow(hWnd);          /* WM_PAINT メッセージをウインドウに送る   */
    return (TRUE);               /* 成功                                    */
}
```

InitApp()

このアプリケーションの他のインスタンスが動作していない場合に，InitAppが呼ばれる．ウィンドウクラスとは作成するウィンドウに関する説明書のことで，WNDCLASS構造体に必要な情報を記入した後に，RegisterClass() 関数でシステムに対して登録する．CreateWindow() との違いはすべてのインスタンスで情報を共有することにある．

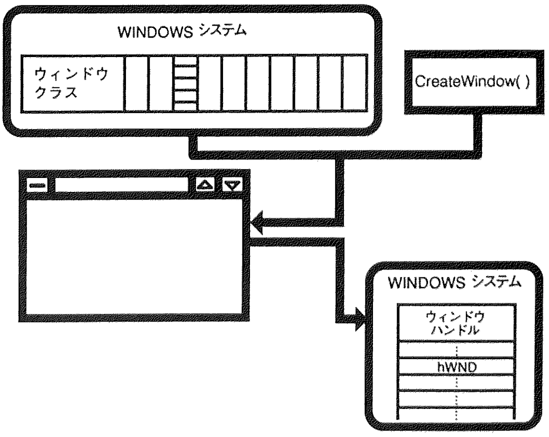


InitInst ()

インスタンスハンドルを保存し，メインウィンドウを作成するのがInitInst() の役割だ．インスタンスハンドルはインスタンスごとに違うので，ここで保存する必要がある．

CreateWindow()

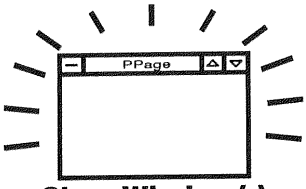
クラス登録されたウィンドウを，CreateWindow() 関数で実際にメモリ上に作成する．CreateWindow() では登録された名前，タイトル，ウィンドウの種類，位置と大きさ，そのほかを指定する．RegisterClass() とCreateWindow() は切っても切れない仲であり，双方が協力して1つのウィンドウを作り上げるのだ．CreateWindow() のリターン値は，ウィンドウハンドルであり，以後はこのウィンドウハンドルを使用して作成したウィンドウにアクセスすることになる．



Windowsプログラミング用語
ハンガリー記法（Hungarian notation）　短い小文字のデータ型名を変数名の先頭につけ，変数名からデータ型を判断できるようにした記名法

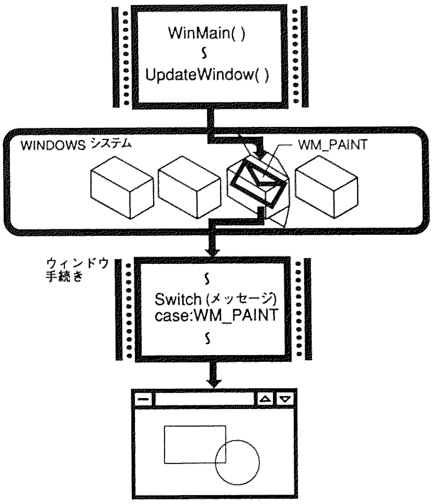
ShowWindow()

CreateWindow() を実行した時点では，ウィンドウは作成されども表示されずといった状態になっている．ShowWindow()関数を実行すると初めてウィンドウを表示する．



UpdateWindow()

WM_PAINTメッセージをWndProc() に送るのがUpdateWindow()の役目である．PPageプログラムでは，WndProc() にWM_PAINT処理ルーチンがないため意味を持たないが，WM_PAINTメッセージの処理ルーチンを持つアプリケーションでは，この命令によって初めてウィンドウ上で描画が始まるようになっている．



COLUMN

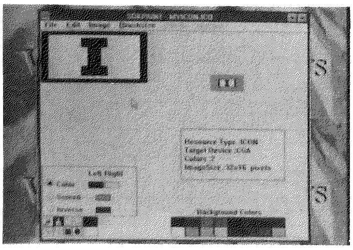
Windows SDKに付属するツール類 ①リソース作成ツール

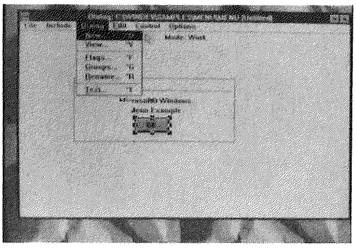
Windows 3.0には，3種類のリソース作成ツールが付いてくる．SDKペイント，ダイアログエディタ，フォントエディタである．

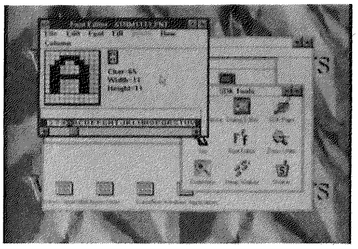
Windows 3.0のSDKペイントは感動ものだ．前のバージョンでは，アイコンやマウスカーソルの作成にはアイコンエディタICONEDIT.EXEを使用していたのだが，あまりにも使いにくくそれはもう涙ものであった．新たに生まれ変わったSDKペイントは，操作性が大幅に向上したのに加え，四角や円を描いたりする機能があったり，パレットが用意されているなど機能的にもアップしている．

SDKペイントを支援するツールとしてZOOMIN.EXEユーティリティがある．これは，ウィンドウ上のビットマップを虫眼鏡で覗くようにズームインするユーティリティである．このZOOMIN.EXEを使えばSDKペイントやフォントツールで画面上のイメージを楽にコピーすることができる．

ダイアログエディタも3.0でパワーアップした．新たなコントロールに対応したばかりでなく，使いやすさを追求している．たとえば，前のバージョンではメニューから毎回コントロールをセレクトしていたのだが，右上にツールボックスとしてアイコン表示を行うことにより，楽にコントロールを扱うことができる．そのほかにも様々な工夫がしてあり，ダイアログボックスの作成が楽しくなるように設計されている．


SDKペイントでアイコン作成


ダイアログエディタ


フォントエディタ

バンディング（Banding）　バンドと呼ばれる矩形の連続でグラフィックイメージを表現する方法．大きなメモリをもたないラスタプリンタでも印刷が可能．

ppage.c ……………………3

```
/******************************************************************
    Window message processing procedure
******************************************************************/

long FAR PASCAL WndProc(hWnd, message, wParam, lParam)
HWND hWnd;                          /* ウインドウハンドル           */
unsigned message;                   /* メッセージタイプ             */
WORD wParam;                        /* メッセージの内容             */
LONG lParam;                        /* メッセージの内容             */
{
    FARPROC lpProcAboutProc;        /* "About" 関数へのFARポインタ  */

    switch (message) {
      case WM_COMMAND:              /* メニューから送られたメッセージ */
        if (wParam == IDM_ABOUT) {
          lpProcAboutProc = MakeProcInstance(AboutProc, hInst);

          DialogBox(hInst,          /* 現在のインスタンス           */
            "AboutBox",             /* リソース名                   */
            hWnd,                   /* 呼び出す側のハンドル         */
            lpProcAboutProc); /* About()のアドレス                  */

          FreeProcInstance(lpProcAboutProc);
          break;
        }
        else                        /* デフォルトの動作をする       */
          return (DefWindowProc(hWnd, message, wParam, lParam));

      case WM_DESTROY:              /* ウインドウ消去のメッセージ   */
        PostQuitMessage(0);
        break;

      default:                      /* アプリケーションが何もしない場合 */
        return (DefWindowProc(hWnd, message, wParam, lParam));
    }
    return (NULL);
}
```

WndProc()

ウィンドウ手続きはWindowsシステムを介して呼ばれるのだが，ここでは先に述べたようにswitch文によりメッセージの内容を解釈して実行する．

ウィンドウ手続きであるWndProc()はlong型の戻り値をもつFARポインタPASCAL型の手続きである．FARポインタとして定義されるのは，プログラムセグメントの外にあるWindowsシステムからコールバック形式で呼ばれるためである．また，PASCAL型の引数引渡し形式が使われるのは，メモリ節約のためでWinMain()の場合と同様である．

引数はhWnd, message, wParam, lParamの4つだ．hWndはウィンドウハンドルであり，WndProc()で処理するウィンドウにはこれでアクセスする．messageはメッセージそのものである．メッセージはWM_DESTROYと

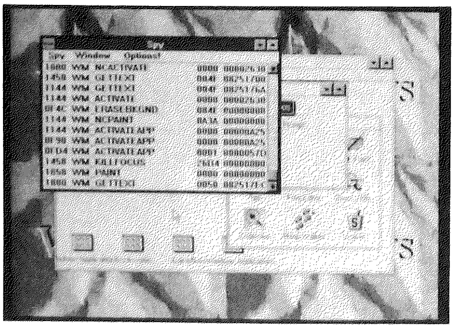
COLUMN

Windows SDKに付属するツール類 ②デバッグツール

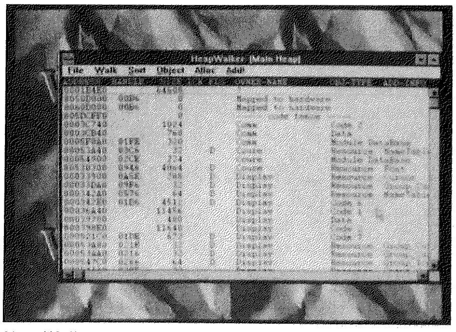
SPYユーティリティは，指定したウィンドウに送られてくるメッセージを解析することができる．まずはWindows勉強用に最適だ．どんなメッセージがウィンドウに送られているのかが克明に表示されるので，メッセージシステムの理解には必須のツールだといえるだろう．デバッグ時にも大きな威力を発揮する．複雑なプログラムになると，ウィンドウ相互間で多くのメッセージがかわされるが，SPYを使えば内部のやりとりが実に鮮明に浮かび上がってくる．

HeapWalkerは，Windowsのグローバルヒープを見るためのツールだ．でも，私はこれをメモリ圧迫機として見ている．徐々に使用メモリを増やしていって，残りメモリ容量が少ない状態をエミュレートできるのだ．こうしてメモリ環境がきつくなった時点で，はじめてグローバルヒープの散歩が役にたつのである．HeapWalkerをただのグローバルメモリ散歩機だと思っている人は，もう少し遊んでみる必要があるかもしれない．

Shakerは移動可能指定されたメモリブロックを，ごちゃまぜにメモリ内で移動させるユーティリティだ．メモリブロックがどんなに移動しても，アプリケーションがちゃんと動くかテストするためのものである．



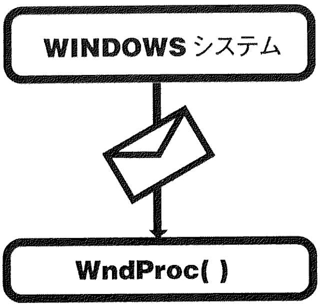
SPY



HeapWalker

Windowsプログラミング用語
ハンドル（Handle）　ハンドルは16ビットのunsigned int型の値で，Windowsが内部参照するために使う数値

かWM_PAINTという形式で記述されるが，それ自身はunsigned int型の数であり，Windows.hファイルのなかで定義されている．wParam，lParamはメッセージに付属するパラメータでそれぞれ16ビット，32ビットの数を保持できる．たとえば，マウスメッセージであるWM_MOUSEMOVEではlParamに現在のXY座標が入るし，キーボードメッセージであるWM_CHARではwParamに押されたキーのアスキーコードが入る．



メニューの処理

WM_COMMANDメッセージは，メニューから送られてくる．メニューそのものは，リソース記述ファイル（ここではppage.rc)に記述されている．リソース記述ファイルではメニュー定義の先頭に

PPageMenu MENU

と書かれている．これは"PPageMenu"という名前のメニューを，リソースとして登録するという意味になっている．では"PPageMenu"という名前がどこに使われているかというと，クラス登録のとき，wc.lpszClassName = "PPageMenu"として登録されるのである．こうしておくことにより，ウィンドウ表示時に自動的にメニューも表示される．また，ユーザーがメニュー項目をセレクトした際には，クラス登録されウィンドウに，メニュー項目のID番号をwParamとしたWM_COMMANDメッセージが送られる．

PPageの場合，リソース記述ファイル内のメニュー定義でIDM_ABOUTを設定しているのでWndProcでIDM_ABOUTを検知し，Aboutダイアログボックスを表示させることとなる．

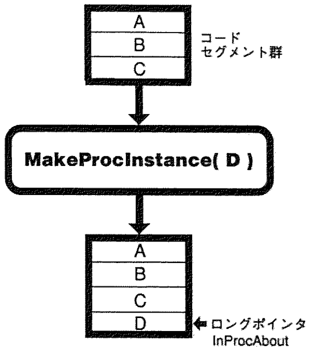
ダイアログボックスの表示

ダイアログボックスもメニューと同様にリソース記述ファイル内で定義される．ダイアログボックス表示で特殊なのは，表示のため新たにインスタンスを作成しなければならないことである．また，ダイアログボックスのインスタンスはダイアログボックス表示前に作成され，表示後ただちに削除されなければならない．

ダイアログボックス表示中は，ダイアログボックスプロシージャにダイアログボックス関連のメッセージが送られる．PPageではAboutProc()がダイアログボックスプロシージャにあたる．

MakeProcInstance()

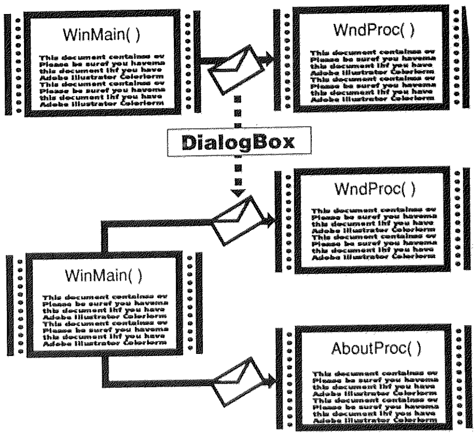
MakeProcInstance()は，ダイアログボックス表示用のインスタンスの作成，すなわちコードセグメントの作成をおこなう．PPageではAboutダイアログボックスのコードセグメントをメモリ上に確保し，その位置をロングポインタlpProcAboutで返すことになる．



DialogBox()

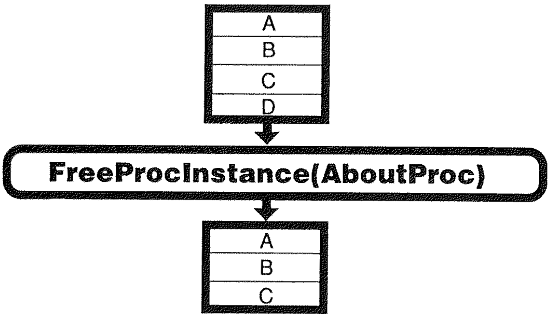
DialogBox()は，MakeProcInstance()で作成されたコードセグメントと，WinMain()にあるDispatchMessage()を結び付ける役割を担っている．DialogBox()が実行されると，メッセージがWndProc()とAboutProc()の双方に分配されるようになる．

また，通常のウィンドウにおける，RegisterClass()，CreateWindow()，ShowWindow()，UpdateWindw()の動作を自動的に行ってくれる．



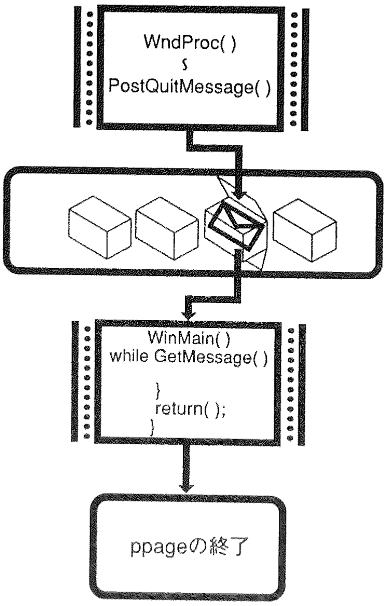
FreeProcInstance()

FreeProcInstance()はちょうどMakeProcInstance()と逆の動きをする．すなわち，MakeProcInstance()で確保されたコードセグメントを破棄するのである．こうして，必要なときに必要なだけメモリを確保することにより，効率的なメモリ運用を実現しているのだ．



PostQuitMessage()

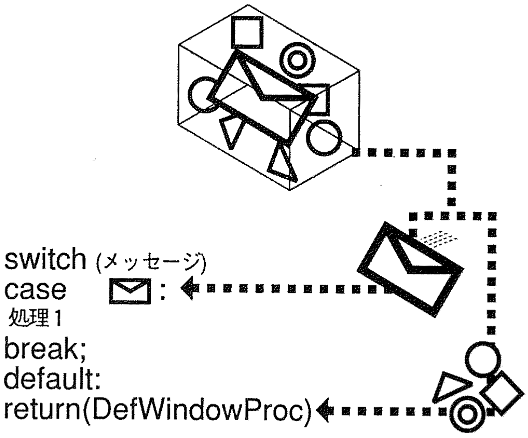
WM_DESTROYメッセージはユーザーがウィンドウのクローズを要求したときにWindowsから送られてくる．次の行のPostQuitMessage(0)；というのはWM_QUITメッセージをメッセージキューのなかにいれよという関数である．WM_QUITメッセージをWinMain()のメッセージループが受け取るとそこでアプリケーションは終了する．while(GetMessage())において，GetMessage()が偽の値を返すからである．



DefWindowProc()

Windowsではすべてのメッセージがswitch()文の中で処理されなければならない．これを実現させるためにWindowsはデフォルト処理用のDefWindowProc()関数を用意している．DefWindowProc(hWnd, message, wParam, lParam)；として使用する．アプリケーションで処理したいもの以外はすべてWindowsシステムにまかせる．

PPageではWM_COMMAND:とdefault:に使用されている．WM_DESTROYには使用されていない．したがって，WM_DESTROY以外はすべてデフォルト処理をするようになっている．



COLUMN

Windows SDKに付属するツール類 ③プログラム最適化ツール

Profilerはどの関数がCPUタイムを無駄に使っているかテストするツールだ．これはWindows上でアイコン起動するだけでOKという他のツールとちがい，プログラムリスト内にうめこんで使うためのツールである．

Swapは無駄なスワップをいかにして減らすかという目的で使われるツールである．モジュールプログラミングの場合，無駄なスワップは大幅な速度低下を招くこととなる．そこで，スワップの頻度を計り，より高速な環境を目指すというのががSwapを利用する目的だ．

Windowsプログラミング用語
ピクセル（Pixels） 画素を意味し，画面表示の最小情報単位．

ppage.c ····················4

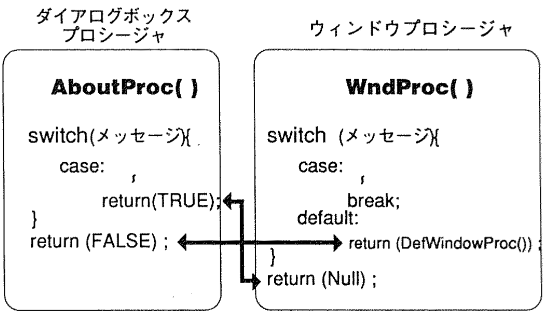
```
/*************************************************************************
    About box message processing procedure
*************************************************************************/

BOOL FAR PASCAL AboutProc(hDlg, message, wParam, lParam)
HWND hDlg;                          /* ダイアログボックスのウインドウハンドル */
unsigned message;                   /* メッセージの種類                       */
WORD wParam;                        /* メッセージの内容                       */
LONG lParam;
{
    switch (message) {
        case WM_INITDIALOG:                     /* 初期化のためのメッセージ   */
            return (TRUE);

        case WM_COMMAND:                        /* ユーザー入力を受け取った   */
            if (wParam == IDOK                  /* "OK"ボタンが押されたのか？ */
                || wParam == IDCANCEL) {        /* クローズコマンドが送られたか */
                EndDialog(hDlg, TRUE);          /* ダイアログボックスを抜ける */
                return (TRUE);
            }
            break;
    }
    return (FALSE);                             /* メッセージが処理されなかった */
}
```

ウィンドウプロシージャとダイアログボックスプロシージャ

PPageではAboutProcがダイアログボックスプロシージャとなっている．ダイアログボックスプロシージャといっても基本的な考え方はWndProc()などのウィンドウプロシージャと変わらない．送られてくるメッセージのうち必要なものを取り出し，必要でないものをデフォルト処理させるのだ．

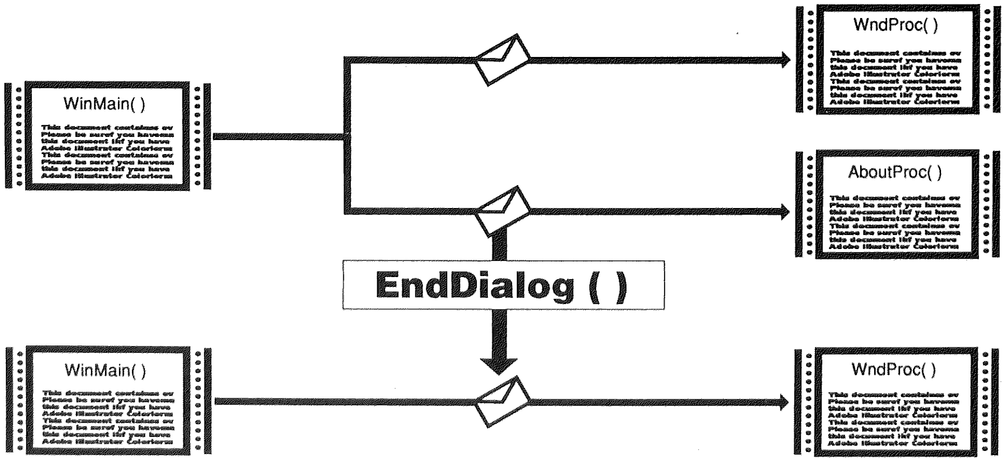
ただし，表記方法が多少異なる．ダイアログハンドルとしてhWndのかわりに，hDlgを用いる．機能的にはhWndと同等とみてよいだろう．したがって，hDlgを使用してタイマーメッセージを送ることもできる．

また，ダイアログボックスプロシージャでは，DefWindowProc()は使用しない．そのかわりに，リターンコードでデフォルト処理の依頼をする．return(TRUE)はWndProc()におけるreturn(NULL)と同等であり，システムにデフォルト処理をさせない場合に使用する．これとは逆にreturn(FALSE)はreturn(DefWindowProc());と同等で，システムにデフォルト処理をさせる場合に使用する．



EndDialog()

EndDialog()はWndProc()におけるDialogBox()と反対に，ダイアログボックスをクローズする役目を負っている．したがって，EndDialog()実行後は，WinMain()からのメッセージがWndProc()のみに送られるようになる．また，終了処理を全部肩代りしてくれる．



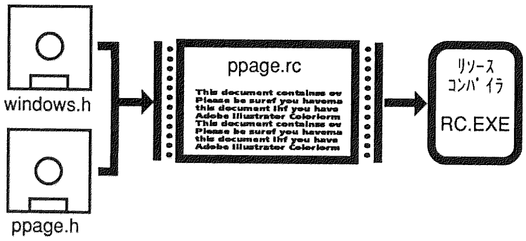
Windowsプログラミング用語
ビットマップ（Bitmaps）　アイコンやラジオボタンなど，Windowsの各構成要素の情報を画面上に表示する基となる画像データ．

ppage.rc

```
#include "windows.h"
#include "PPage.h"

PPageMenu MENU
BEGIN
    POPUP       "&Help"
    BEGIN
        MENUITEM "&About PPage...", IDM_ABOUT
    END
END

AboutBox DIALOG 22, 17, 144, 75
STYLE DS_MODALFRAME | WS_CAPTION | WS_SYSMENU
CAPTION "About PPage"
BEGIN
    CTEXT "Microsoft Windows"      -1,     0,  5, 144,  8
    CTEXT "Programmer's Page Application"   -1,     0, 14, 144,  8
    CTEXT "Version 3.0"            -1,     0, 34, 144,  8
    DEFPUSHBUTTON "OK"          IDOK,     53, 59,  32, 14,      WS_GROUP
END
```

リソースファイルの設定

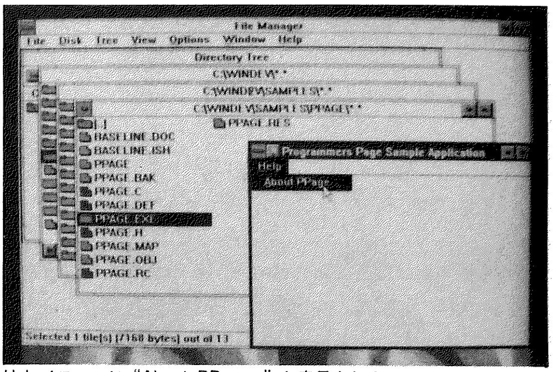
リソース定義ファイルにおいても，windows. hとppage. hをインクルードする．メニュー定義におけるIDM_ABOUTはppage. hで定義されている．



メニューの定義

PPageMenuという名のメニューリソースを定義している．メニュー定義はBEGINで始まりENDで終了する．POPUPというのはタイトルバーの下の段に表示されるメニューのことで，PPageMenuでは"Help"と表示される．また，POPUPの下のBEGIN，ENDにはさまれたMENUITEMは，"Help"が選択されたときのメニュー内容で，"About PPage..."と表示される．もし，"About PPage..."が選択されると，IDM_ABOUTをパラメータとするWM_COMMANDメッセージが，WinMain()におくられる．

ちなみに，&Helpの先頭についている&マークは，コントロールキーによる操作をあらわし，このばあいは，[Ctrl]+[H]で，マウスでクリックしたとおなじ効果を実現できる．



Helpメニューに "About PPage..." と表示される

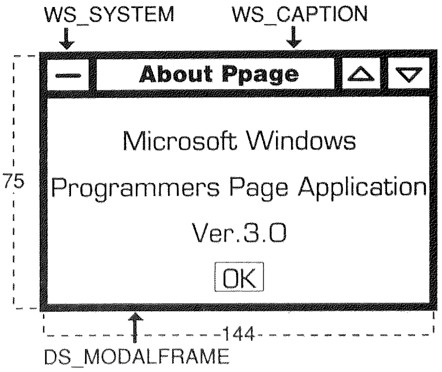
ダイアログボックスの定義

AboutBoxという名のダイアログボックスを定義している．位置が（22, 17）で，横144，縦75のサイズのダイアログボックスが描かれる．

スタイルとしてMODALFRAME型のフレームで，キャプション，システムメニューつきメニューが定義される．キャプションの内容は"About PPage"である．

BEGINからENDまでで，ダイアログボックスの内容が決まる．CTEXTとは，中央ぞろえ書式のテキスト表示設定，DEFPUSHBUTTONはデフォルトのプッシュボタンを意味する．パラメータとして，左から，IDナンバー，横の位置，縦の位置，横の幅，縦の幅，オプションとなっている．

ダイアログエディタを使用すれば，高度なダイアログボックスが作成できる．



Windowsプログラミング用語
ビューポイント（Viewpoint）　マッピングは物理座標でおこなわれ，たいていの場合ビューポイントはクライアントエリアと一致する．

ppage.def

```
; Programmer's page module-definition file

NAME        PPage

DESCRIPTION  'Programmers page Application'

EXETYPE      WINDOWS

STUB         'WINSTUB.EXE'

CODE  PRELOAD MOVEABLE DISCARDABLE

DATA  PRELOAD MOVEABLE MULTIPLE

HEAPSIZE     1024
STACKSIZE    5120

EXPORTS
        WndProc       @1
        AboutProc     @2
```

モジュール定義ファイル

モジュール定義ファイルとは，リンク時に参照するテキストデータファイルのことで，拡張子が.DEFのファイルがこれにあたる．最初のうちは存在を忘れがちであるが，大きなプログラムを組む際には非常に重要になってくる．

まず，NAME文でモジュールの名前を設定する．通常はファイル名をあてる．DESCRIPTION文にはモジュールの説明を記入する．この内容はプログラムマネージャ上でPPageをアイコン化した際にアイコンの下に表示される．次にEXETYPE文であるが，Windowsアプリケーションということを識別するために必ずWINDOWS指定をしなければならない．

STUB文にはDOS上からPPageが起動された場合に実行されるファイル名を記入する．WINSTUB.EXEは，Windowsが必要な旨を表示して終了するだけのプログラムである．もし，ここを他のDOSプログラム，たとえばCHKDSK.EXEと記入すると，Windows上で起動するとPPage，DOS上で起動するとCHKDSKが実行される．このしくみを使えばDOS，Windows両用ツールなどというものもできるかもしれない．

CODE文ではコードセグメントの属性を定義する．Windowsはメモリの有効活用のため386エンハンストモードの場合を除き，セグメントの移動・廃棄を随時繰り返している．コードセグメントが移動可能なときはMOVEABLEを，破棄可能なときはDISCARDABLEを指定する．

DATA文は，データセグメントの属性を定義する．複数インスタンスで起動される場合はMULTIPLEにする必要がある．MULTIPLEにすると1つのインスタンスにつき1つのデータセグメントが割り振られる．

HEAPSIZE文はアプリケーションに必要なローカルヒープのバイト数を定義する．STACKSIZE文はアプリケーションに必要なローカルスタックのバイト数を定義する．

最後にEXPORTS文であるが，Windowsから呼ばれる可能性のあるユーザー関数はすべてここに記載しなくてはならない．PPageではWndProcとAboutProcがこれにあたる．これらはWindowsからメッセージとともに呼び出される．通常は，ウィンドウプロシージャとダイアログボックスプロシージャについて気をつけておけば十分だと思う．最初のうちはEXPORTSに記載するのを忘れて暴走を繰り返すかもしれないがじきに慣れるだろう．

Windowsプログラミング用語
フォーカス（Focus）　複数のウィンドウの中からキーボードからのメッセージを受け取る権利のあるウィンドウをフォーカスのあるウィンドウという．

ppage

```
# Programmer's Page sample program make file.

all: ppage.exe                                    ①

# Update the resource file

ppage.res: ppage.rc ppage.h                       ②
    rc -r ppage.rc

# Update the object file

ppage.obj: ppage.c ppage.h                        ③
    cl -c -AS -Gsw -Oas -Zpe ppage.c

# Update the executable file and add the resource

ppage.exe: ppage.obj ppage.def                    ④
    link /NOD ppage,,, libw slibcew, ppage.def
    rc ppage.res

# Add the resource

ppage.exe: ppage.res                              ⑤
    rc ppage.res
```

●

①この1文はMS-C5.0用のメイクファイルをMS-C6.0で利用する場合に必要.

②ppage.hまたはppage.rcが変更されていたら，ppage.rcをリソースコンパイラ（rc）を用いてコンパイルし，ppage.resを作成する.

③ppage.hまたはppage.cが変更されていたら，ppage.cをCコンパイラ（MS-C）を用いてコンパイルしppage.objを作成する.

④ppage.objまたはppage.defが変更されていたら，モジュール定義ファイルppage.defの情報に基づいて，リンカ（LINK）を用いてppage.objをライブラリとリンクし，ppage.exeを作成する. さらにppage.resをリソースコンパイラを用いてppage.exeにコピーする.

⑤ppage.resが変更されていたらppage.resをリソースコンパイラを用いてppage.exeにコピーする.

MAKEファイル

MAKEは図17のように行う. 注目すべきはリソース関連とコンパイラ関連が完全に独立していることである. このことにより，リソースのみの修正，コンパイルソースのみの修正などが容易にできる.

MAKEファイルの書式では，コロン（：）の左側のファイルの作成（更新）日時が右側のファイルよりも古い場合に真となり，次の行に書かれた命令が実行される. たとえば

```
ppage.res: ppage.rc ppage.h
    rc -r ppage.rc
```

はppage.rcまたはppage.hのどちらかのファイル日付がppage.resよりも新しい場合には次の命令 rc -r ppage.rcを実行するという意味である.

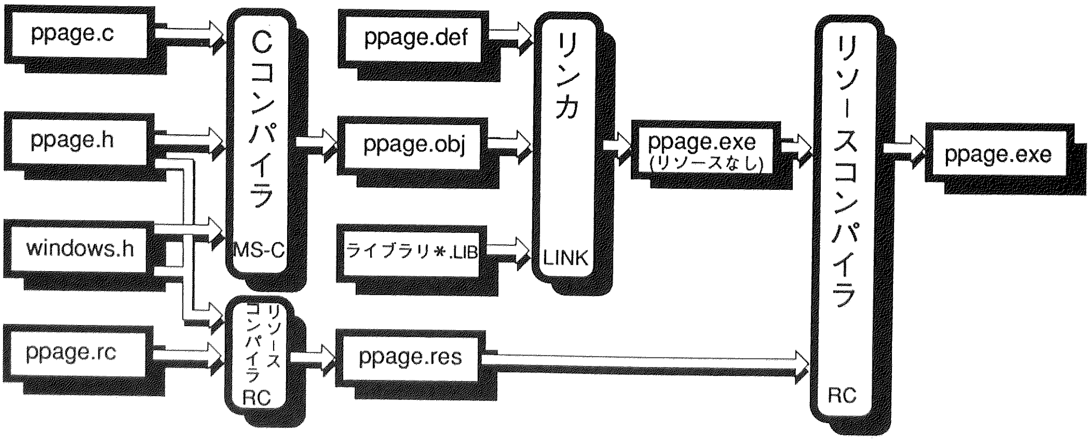
●

ppage.cのみを修正したときは，ppage.rcは更新されていないので，リソースのコンパイル（②）は行われずに，ppage.cのコンパイル（③），リンクが実行されたあと，ppage.resがコピーされる（④）. この場合にはppage.exeの方がppage.resよりも後から作成されているので，⑤は実行されない.

ソースはそのままでリソース定義ファイルppage.rcのみを修正したときは，まずリソースのコンパイルが実行される（②）. ppage.obj，ppage.exeは前回のままなのでppage.cのコンパイル（③）リンク（④）は行われないが，ppage.resが再作成されたのでppage.exeにコピーされる（⑤）.

さいごに

以上，PPageのファイル群について，大解剖を行ってきた. 最初に言ったように，Windowsのプログラミングが上達するためには，スケルトンを元にして肉付けしていくのが，いちばんの近道である. DOS上でのプログラミングに慣れた人には，なかなかとっつきにくいと思うが，がんばって挑戦してほしい.



Windowsプログラミング用語
ブラシ（Brushes）　ビットマップ化されたピクセルパターンを定義するGDIの関数で，閉じた領域の塗りつぶしなどに利用される.

Windows
プログラミング講座
4

実践的Windows プログラミング技法

イースト株式会社 取締役研究開発部長
滝口 力

イースト株式会社は，WindowsやOS/2プレゼンテーションマネージャのプログラングでは定評のあるソフトハウスである．Windows上のワープロソフトText Writerや日本語入力ソフトE1WINなどのパッケージを発売するとともに，多くのWindows関連ソフトウェアを受注開発している．国内ではもっとも早い時期にWindowsプログラミングへの取り組みをはじめただけに，身をもって知った実践的テクニックを蓄積している．ここでは，同社の研究開発部長をつとめる滝口力氏が，DOSプログラムとの考え方の違い，とくに注意すべきこと，後回しにしてもいいことなど，マニュアルからは得られない情報を経験の重みとともに伝えてくれる．Windowsプログラミングに取り組んでいる，あるいはこれから挑戦しようというプログラマによき指針を与えてくれることだろう．

Windowsプログラミングの基本構造

Windowsのプログラムについてまずいえることは，これはもうC言語のアプリケーションではないということです．足し算や掛け算などにはCの関数を使いますが，それ以外の命令を使って何か処理をするときには，ほとんどWindowsの関数を使います．いままでCでプログラムを書いていたプログラマであっても，全然なじみのない関数ばかりを使うことになります．まずはこれを念頭においてください．

それにくわえてWindowsは複数のプログラムが動作するマルチタスク環境であること，ウィンドウシステムという独特のインターフェースを持っていることから，MS-DOS上でのC言語のプログラムとは異なった構造を持っています．まずはそこから話を始めましょう．

ADVICE1
Cのプログラムだと考えるな！

メインボディとウィンドウプロシージャ

Windowsのプログラムの一番の基本はメインボディとウィンドウプロシージャ，この２つです．C言語でプログラムを書かれた方はわかると思いますが，C言語にはひとつだけ必ず書かなくてはならないものがあります．それはmainです．それ以外の関数は自分で勝手な名前をつけていくらでも書いていけますが，mainというプロセスは絶対に書かなくてはなりません．Windowsの場合もそれとまったく同じで，メインボディーWinMainという名前になっています－これが最初に必ず起動されます．ウィンドウプロシージャについては名前は自分でつけられます．

Windowsというのは，基本的にメッセージによって動きます．動くというとわかりにくいのですが，ユーザーの作ったアプリケーションがあるとするとここにWindowsが必ずメッセージをくれます．

どういうことかといいますと，パソコンにはキーボードがあったり，マウスがあったり，CRTがあったり，プリンタがあったりするわけですが，これらの装置から何か入ってきますね．ユーザーのアプリケーションプログラムは自分では入力を拾いません．必ずWindowsが拾って「いま何をされた」ということをメッセージの形で伝えてきます．ですから，複数のアプリケーションが動いていても，いまキーを入力したのはどのアプリケーションに対してかということはWindowsが認識していますので，正しくアプリケーションに伝えられます．ですから，Windowsのアプリケーションというのは基本的に自分で入出力はしません．プログラムを書く人は自分で読んだり書いたりするための手続きを書いていますが，Windowsの場合は基本はこの形です（図１）

イベント駆動型

OS/2ではOS/2というオペレーテ

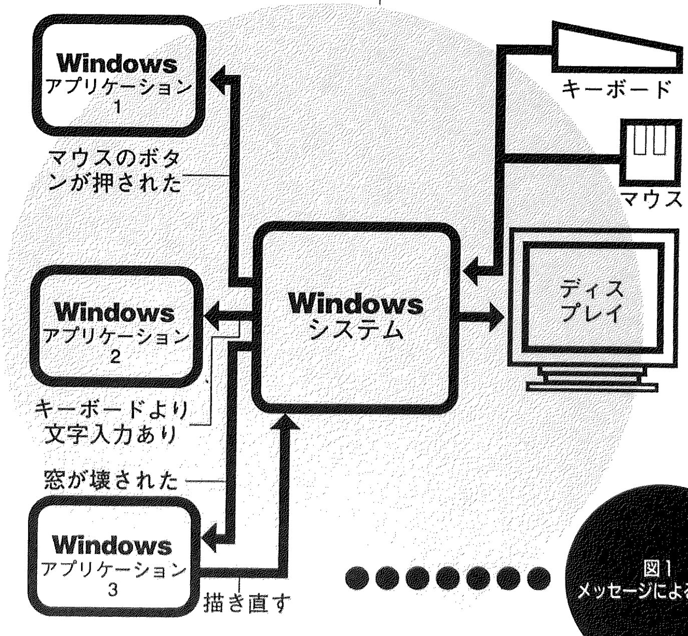


図1
メッセージによる入出力

Windowsプログラミング用語
プリエンティブ（Preemptive (multitasking)）　マルチタスクの先取り権のこと．Windowsはイベント駆動型のノンプリエンティブなマルチタスクである．

ィングシステムの上にプレゼンテーションマネージャが載っていてMMI(注1)を司っています．Windowsもそれとまったく同じようなことをしていますが，ひとつ決定的な違いがあります．OS/2というのは完全なリアルタイムのマルチタスクOSですが，MS-DOSというのはそうではありません．シングルタスクOSです．Windowsはその上で疑似的なマルチタスク環境を提供しています．マルチタスク環境であるということが，必然的に「イベント駆動型」という形のプログラムを作ることを強制しています．

2つの処理を同時に動かそうとすると，一方が動いているときに，もう一方をどういうタイミングで動かすかという問題になってきます．通常のマルチタスクOSであれば，非常に複雑な制御をしています．一定時間一方が動いたら，他方を動かすとか，一方がある宣言をしたら他方を動かすとか，2つの処理の間でやりとりをして，交互に動かしたりするなど，いろいろなことをやっています．

Windowsの場合は，非常に単純な方法を採用しています．プログラム1がWindowsに何かを要求をしたときに，強制的にプログラム2に切り替えます．プログラム2がWindowsに対して何かを要求したときには，またプログラム1に切り替えます．ですからプログラム1がずっとWindowsに何も要求を出さないと永久にプログラム2は動きません（**図2**）．

このような方式になっているところから，イベント駆動型という構造をとっているわけです．これがプログラムをとても作りにくくしています．先ほどお話したWindowsのメインボディというのはCRT上に窓を開くだけです．実際の処理はウィンドウプロシージャのほうで行います．ウィンドウプロシージャにはWindowsからどんどんメッセージがきます．このメッセージを処理したら，もう一度Windowsに制御を返さなければいけないわけです．メッセージがきて何か処理をしている間はそのプログラムがアクティブです．Windowsに制御がもどるとWindowsはほかに動かすべきものがあれば，そちらに制御を渡します．

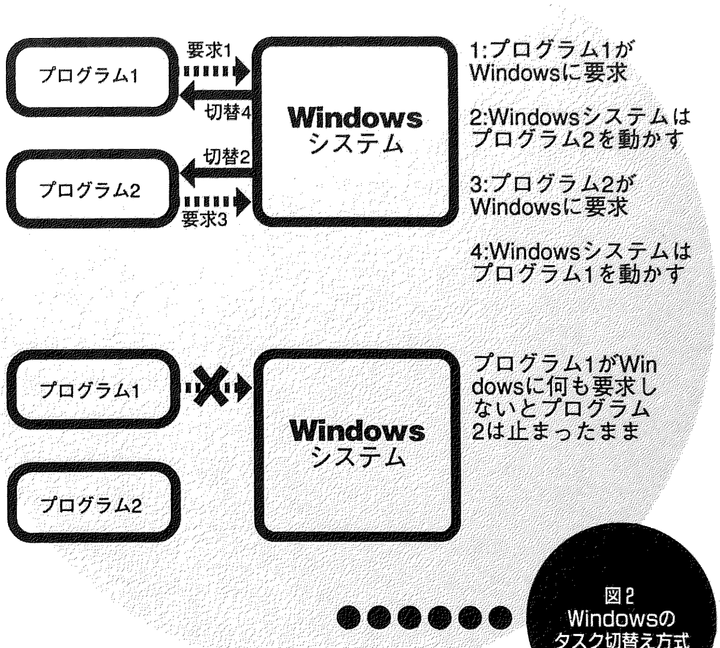
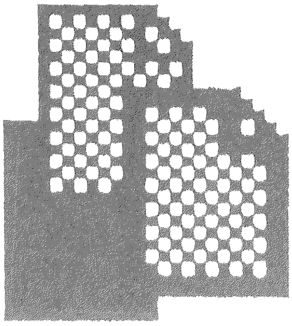


図2 Windowsのタスク切替え方式

COLUMN

構造化プログラミング

ダイクストラ（E.W.Dijkstra）によって提唱されたプログラミングの方法論．構造化プログラミングではシステムの構造を1機能が1モジュールとなるよう機能分解する．3つの基本となる制御構造「連接」（Sequence）「判断」（if then else）「反復」（while）を組み合わせてプログラムを作成する．これによって，生産性を高め，goto文のない読みやすいプログラムを作成しようとする．



Windowsプログラミング用語
プレフィックス（Prefix (notation)）　ハンガリー記法のように変数名の前につけるデータ型名などをプレフィックスという．

それではプログラムはどのような形式で書いたらいいのでしょう？　非常に昔からあるものにはフローチャートがありますね．これはSTARTからSTOPまで順序性のある構造です．これに対して10年ほど前に構造化プログラミング(STRUCTURED CHART)という方法が出てきました．昔マイコンをやっていた方はわかると思いますが，割り込み処理などをするときにはプログラムというのは非常にバラバラなんです．何かが発生するとある処理をして終わるという塊です．つまり順序というものを無視した，機能の種類ごとに完結した作りです．

Windowsプログラムでも何か処理をしたら必ずWindowsに制御を返さなければならない．したがってこれと同じような形，機能単位の処理というのを採用しているわけです．ある処理で値をセットして，別の処理のとき使うというのはルール違反なんですね．Windowsがマルチタスクであるが故にこういう処理の仕方を強制されるわけです．

PMでも同じ作り方をします．大型機やUNIXの世界でも当り前の話です．ところが，パソコンのシングルタスクの世界でプログラムを作ってきた人には非常に理解しづらいでしょう．

(注1)MMI man-machine interfaceの略．人間とコンピュータの間で情報をやり取りするための仲介役を果たすソフトウェア，ハードウェアのこと．

ペイントプロシージャ

もうひとつ大切なこととして，Windowsはウィンドウシステムであるということがあります．MS-Windowsというのは，ウィンドウシステムとしてはどちらかというとレベルの低いものです．では，高級なウィンドウシステムとの違いはどこにあるのでしょうか．それは複数のアプリケーションが動いたときに見られます．ある窓を開いて処理をしているときに，ユーザーがもう１つ窓を開く．そしてこの後から開いた窓の処理が終ると窓が消えます．すると消えた跡が白く抜けます．これをリドロウ(再描画)処理するわけです．高級なウィンドウシステムでは再描画をウィンドウのシステムがやってくれます．ところがMS-WindowsやPMではユーザーにその部分を描きもどすことを要求しています．これがプログラムを非常にわかりにくくしているもうひとつの原因です（図３）．

Windowsは「窓が壊れた」ということと，同時に壊れた範囲を教えてくれますから，その範囲を描きなおします．プログラムにこの処理をきちんと入れておかないとうまく動きません．プログラマにとっては非常にイヤなことですが，これがペイントプロシージャです．再描画要求そのものは，WM_PAINTというメッセージの形でWindowsからアプリケーションに渡されます．これを実際に受け取るのはメッセージを処理するプロシージャです．再描画処理そのものが独立したプロシージャというわけではありません．

そうするとプログラムの基本構造として３つの要素が必要になります．メインボディとメッセージを処理するプロシージャ，それに再描画をするプロ

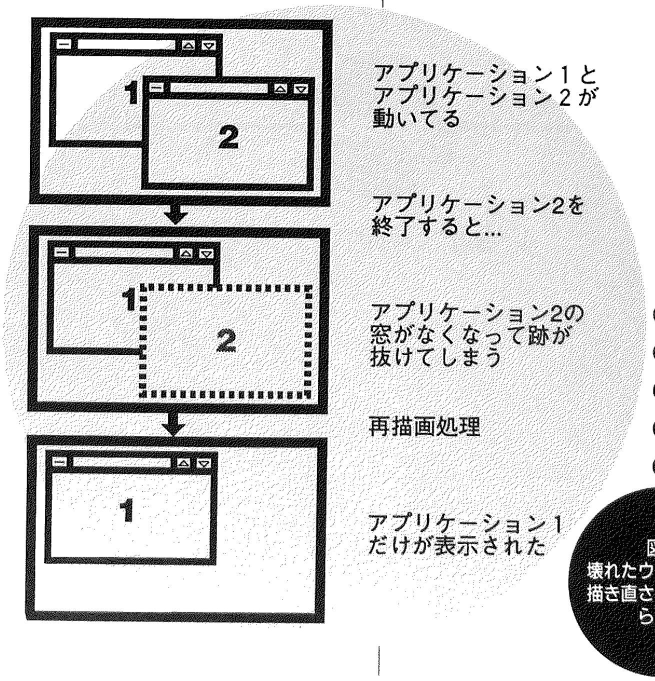


図３
壊れたウィンドウは描き直さなければならない

Windowsプログラミング用語
プロシージャ (Procedure)　手続きや手順という意味．値を返す関数 (function) に対してつかわれる．ブロックやサブルーチンと同じ．

シージャ．この3つが揃って初めてWindowsのアプリケーションとして形をなすわけです．Windowsプロシージャで実際にメッセージを受け取ったときの処理をします．具体的に見るとこれはCASE文のかたまりです．

```
case メッセージ1:
        処理1
          ・
          ・
        break;
case メッセージ2:
        処理2
          ・
          ・
        break;
          ・
          ・
```

もうひとつが再描画です．そしてこの2つがともに従来のDOSプログラマにはなじまないわけです．つまりほとんど全部がなじまないということになってしまいます．これは一見するとWindowsの欠点のように見えてしまいます．

モジュール定義ファイル(.DEF)とリソースファイル(.RC)

Windowsのプログラムには，基本的なファイルが4種類あります．

DOS上のアプリケーションでは，＊.CというC言語で書いたプログラム，人によってはヘッダファイル＊.H，それとは別にコンパイルやリンクをするためのバッチファイル＊.BAT，MAKEファイルがあります．Windowsの場合はそれにプラスして＊.DEF，＊.RCというファイルが必須です（なくても作れないことはありません）．

＊.DEFファイルは，これから作る＊.EXEファイルの性格を示します．＊.EXEファイルのメモリの使い方(Windowsは内部でメモリを動かしますから，基本的にこのアプリケーションはメモリを動かされては困るとか，固定しなくてもいいとか)，それからスタックとして，ローカルヒープエリアとしてどのくらいの量を使うとかいう内容も宣言します．つまり，プログラムの動作環境を設定するのがこの＊.DEFというファイルです．DOSではスタックの指定はLINKのときにしていましたが，Windowsの場合はまったく違うリンカプログラムLINK4.EXEを使います（ただし，MS-C5.1以降では，LINK.EXEでもできます）．

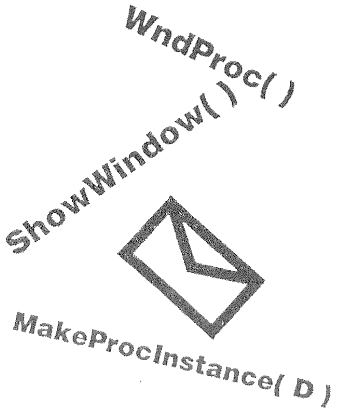
次に＊.RCですが，これはヘッダファイルの発展形みたいなものです．DOSのプログラムとも最近はいろいろなユーザーインターフェースを持っていて，メニューバーが出て，メニューが出たり，パネルが出たりします．下手な人が作るとこういう文字をすべてプログラムの中に書きます．ちょっと賢い人ならヘッダファイルの中に作ります．そうしておけば，メニューやパネルアップデートするときにヘッダファイルだけを直せばすむからです．Windowsではメニューやパネルを別ファイルで定義します．これをリソースといいます．リソースとして定義すると「ボタンが押されるとボタンの色が変わる」「押されたことを認識する」「文字を入力してリターンを押す」ここまでの制御はWindowsがすべてやってくれるという点でヘッダファイルとは違います．リソースはパネルやメニューの文字や形を宣言をするだけなので，基本的なコントロールはアプリケーションから見えないようになっています．つまり，リソースとはパネルやメニューを宣言して，その制御をWindowsに頼むためのものだと理解してください．

リソースをプログラムに組み込むために，Windowsにはリソースコンパ

COLUMN

イベント駆動型とメッセージ駆動型

「イベント駆動型」と同じ意味あいで「メッセージ駆動型」ということもある．アプリケーションとシステムとの間でやりとりするものを何と呼ぶかの違い．MS-Windowsではこれを「メッセージ」と呼んでいる．イベントとはもっと広い意味で「ボタンを押した」ということ自体，つまりマシンの外側あるいは内側（タイマ割り込みなど）で起こった事象を指している．アプリケーションはメッセージを受け取って何かの処理を行うのだが，そのメッセージの元がイベントというわけだ．



ペン（Pens）　WindowsのGDI関数のひとつで画面描画時に使う．あらかじめストックペンが定義されているが，ユーザー独自の色や幅，スタイルが設定できる．

LIST

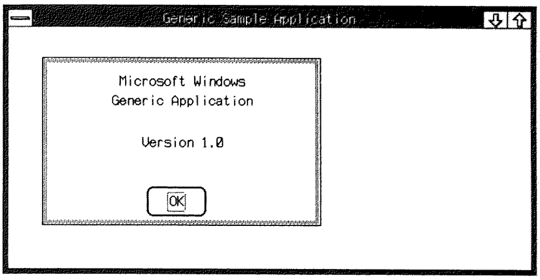
(Windows SDKサンプルより)

モジュール定義ファイル

```
;module-definition file for generic -- used by link4.exe

NAME      Generic         ; application's module name

DESCRIPTION 'Sample Microsoft Windows Application'

STUB      'WINSTUB.EXE'  ; Generates error message if application
                         ; is run without Windows

CODE      MOVEABLE        ; code can be moved in memory

;DATA must be MULTIPLE if program can be invoked more than once

DATA      MOVEABLE MULTIPLE

HEAPSIZE  1024
STACKSIZE 4096   ; recommended minimum for Windows applications

; All functions that will be called by any Windows routine
; MUST be exported.

EXPORTS
    GenericWndProc    @1   ; name of window processing function
    About             @2   ; name of "About" processing function
```

リソースファイル

```
#include "windows.h"
#include "generic.h"

AboutBox DIALOG 22, 17, 144, 75
STYLE WS_POPUP | WS_DLGFRAME
BEGIN
    CTEXT "Microsoft Windows"      -1,      0,  5, 144,  8
    CTEXT "Generic Application"    -1,      0, 14, 144,  8
    CTEXT "Version 1.0"            -1,      0, 34, 144,  8
    DEFPUSHBUTTON "OK"           IDOK,     53, 59,  32, 14,  WS_GROUP
END
```



イラ(RC.EXE)が用意されています．リソースファイル（＊.RC）をコンパイルすると＊.RESファイルができます．次にLINK4という特殊なLINKを使って，＊.DEFファイルと＊.OBJをくっつけて＊.EXEファイルを作ります．さらにもう一度RC.EXEを使って＊.RESファイルと＊.EXEをくっつけます．これでWindowsのEXEファイルの完成です．

したがって，Windowsアプリケーションを日本語化するのは（表示に限っていえば）非常に簡単です．＊.RCファイル中のメッセージを日本語にしてRCとLINK4を使えば，出てくるメニューやパネルは日本語になります．

プログラミング作法

Windowsのプログラミングが従来のDOSでのプログラミングと違うのは，「作法」を強制しているという点です．DOSでのプログラミングには「うまく作ればいい」「動けばいい」というところがあります．一方きれいに書くのが良い，後で誰が見てもわかるように書くのが良いとすることもありますが，これは一種の幻想にすぎない面もあります．プログラムソースというのは寿命が長くありません．同じソースを10年も使ってアップデートを繰り返すことはあまりないことです．今の世の中のスピードに比べたらその財産価値はほとんどないといってもいいでしょう．頭の中にノウハウが残るだけです．だからこそ，よけいに「汚くてもちゃんと動けばいい」という考え方がでてくるのです．

ADVICE3
Windowsプログラムはきれいに書かないと動かない

ところが，Windowsのプログラムはきれいに書かないと動きません．何をやるにもWindowsが提供した関数を使わなければなりません．DOSではグラフィックのVRAMをアクセスできる人というのは優れたプログラマだったわけです．標準の入出力を使って遅いプログラムを組むのではなく，自分で番地をとってきて，そこにデータを詰めていくという技術を持っていることは優れたプログラマの条件でした．ところがWindowsはそういうことがまったくない世界です．その意味で，Windowsは優れたプログラマにとっておもしろくないかもしれません．

Windowsプログラミング用語
ポートレイトモード（Portrait mode）　プリンタで短辺を上にして用紙を入れるモード．Windowsはモードによってプリンタドライバで縦横の解像度を変更する．

このように作法を強制するプログラムは一般的にいっても非常に冗長です．DOSプログラムの基本的なサンプルに「hello」がありますね．「hello」と１行画面に表示するものです．

```
main () {
  printf ("hello¥n") ;
}
```

全部でたった３行ですみます．これをWindowsでプログラミングすると，全部で100行近くなります．メインボディがあって，ウィンドウプロシージャがあって，さらにペイントプロシージャがあるのですから．ただ，基本形が冗長であるということは，本当に大きなプログラムを作ったときは，短くなるかもしれません．また，作法が強制されて，かつ冗長であるということは，メンテナンス性は高いのかもしれないということはいえます．

プログラム開発の4つのポイント

ここまでの説明を終えたところで，みなさんにWindowsプログラミングについての本をポンと渡して，「さあ，組んでください」といっても，まず無理でしょう．おそらく投げ出してしまうと思います．Windowsでプログラムを開発するにはいくつかの大切なポイントがあります．

①サンプルを活用せよ

Windowsのプログラミングを覚えるには，まずサンプルプログラムの変形から入ります．MS-Windows の SDK (注２)を買うとかならずサンプルプログラムがついてきます．サンプルプログラムのメニューのパターンを変えてみようとか，パネルの形を変えてみようとか，ひとつメッセージを追加してみようとか，そういう作業を通して理解していくという方法を取らない限り，Windowsのプログラミングは習得できません．

極端な話，WinMainやウィンドウのプロシージャなどは，私どもが開発するときでも全部サンプルからコピーします．うちの会社には，WinMainというスペルをたたいたことのあるプログラマはいないはずです．窓を開くCreatWindowのX座標とかY座標とか，そういうところだけを直すのです．Windowsを１～２年やったプログラマでも関数の綴りを誤りなしで書くことはできないでしょう．サンプルプログラムには，「ファイルの一覧を出す」「文字を入力する」といった基本的な処理はすべて含まれていますから，これを利用することが成功の早道です．

(注２) SDK　Windows Software Development Kit, Windowsソフトウェア開発キットのこと．メーカーによってはWindowsツールキットという呼び方もある．

ADVICE4
関数を覚えるな！サンプルを使え！

②マニュアルは辞書として使え

Windowsのプログラミングがプログラマになじまないもうひとつの理由は，マニュアルが多いことです．メインフレームをやったことのある人にとっては当り前ですが，IBMのマニュアルが山のように積まれます．Windowsもかなりの量のマニュアルがあります．しかもこれまで持っていたMS-Cの入門書などはほとんど役に立ちません．１文字表示するにもWindowsの関数を見なければいけないのですから．

COLUMN

Windows SDKのマニュアル

現在日本で市販されているWindows2.11用のSDKを買うと，次のマニュアルがついてくる．

1．MS-Windows セットアップ／アプリケーションスタイルガイド
2．MS-Windows プログラマーズラーニングガイド
3．MS-Windows プログラミングツールズ
4．MS-Windows プログラマーズリファレンス Vol.1
5．MS-Windows プログラマーズリファレンス Vol.2
6．MS-Windows エクステンションズ

Windowsプログラミング用語
ホットスポット（Hot spot）　マウスがディスプレイ上で示す座標のこと．通常はマウスカーソルの矢印の先端や十字の中心が示す座標がホットスポット．

マニュアルは辞書として使うこと．まずスケルトンプログラムの改造から入って，わからない言葉がでてきたら調べるという形で使います．

③細かな理解は後回しで

第3のポイントは細かいことの理解は後回しにするということです．

Windowsというのは OSのことを非常によく理解している人が考えたのだろうという気がします．ハンドルやオブジェクトといった考え方はパソコンに向かってばかりいる人の頭の中からは出てこない，きわめてアカデミックなものです．しかし，これらの考え方を徹底的に理解するまで待っていたのではなかなかプログラミングできません．

Windowsでは，ファイル，メモリ，ディスプレイなどの部分はシステムが吸収するという考え方をとったために，きわめて論理的な世界になっています．したがって，ファイルやメモリやディスプレイなど，ひとつひとつに論理的な値を与える必要があります．これをハンドルといいます．あるファイルに書き込みたいとすると，まずはそのファイルのハンドルを手に入れます．メモリをとると，そのメモリのハンドルが返ってきます．そしてそのハンドルを使ってメモリを返したり，何かを書き込んだりします．

オブジェクトというのは，ある「意味のあるもの」たとえば文字であるとか，イメージであるとか，そういう意味のあるものを作ってとっておくときに使います．線を一本引くときには，「太さ1のペン」を使います．これがオブジェクトです．太さ2の線を引くときは，「太さ2のペン」というオブジェクトを作らなくてはなりません．ペンを使うときはSELECTします．オブジェクトは何個も作って持っていることができますが，一度に使えるのはひとつだけです．別の太さの線を引きたいときは，いま使っているペンをいったん返して，SELECTし直せばいいのです．

文字を1つ出すにはTextOutという関数があります．描画するにはBit-Bltという関数があります．こういう関数では座標の位置などとは別に，あるハンドルを指定することができます．ディスプレイに出すのであれば，ディスプレイコンテキストというハンドルを指定します．

実行前に必ずハンドルを確保しなくてはならないのです．つまり，あるひとつの関数を実行するときには必ずサンドイッチを作ります．ハンドルやオブジェクトを選んだり取ったりして，それから実行して，終ったら返します．

こういう処理を知らない人が見ると何をやっているのかさっぱりわかりません．関数のマニュアルを見れば何をやればいいか，どの関数を呼べばいいか，その1行はわかります．でも，その前後には必ずおまじないがつくわけです．GetObject xxx, SelectObject xxx，これが理解を妨げています．

きちんと理解していなくても真似をして書いていればたいていうまくいきます．だからこそ最初にやるときは，細かいことにはこだわらないことです．こうやればうまくいくとわかっていたら，その理由は知らなくてもかまいません．うまく行く通りやっておけばいいのです．理由はそのうちだんだんとわかってきます．

細かいことにこだわるなというのは，そういう意味です．理解して納得するまで何もできないというような人にはWindowsのプログラミングは向いていません．

④メモリがきついので注意を要する

メモリの制約は，Windows2. x最大の欠点です．これは，なまじっかなきつさではありません．カナ漢字変換とプリンタドライバを組込んだシステムがあるとしましょう．その状態でWindowsを立ち上げたときに使えるメモリは200 KBありません．かなりのメモリをWindowsのシステムが占有してしまうわけです．その上にさらにアプリケーションが入ってくるとなると残りはいくらもないことは明白です．

慎重に計算してやればうまくいくだろうと思うと，そうはいきません．Windowsシステムはかなり勝手にメモリを使います．しかもアプリケーシ

Windowsプログラミング用語
マッピングモード（Mapping modes）　プログラムから送られた論理座標値をマッピングモードにあわせて変換しスクリーン上に表示する．

ョンを作っている人にはなかなかわからないのです．再描画の話にもどりますが，Windowsの画面でメニューやパネルを表示して元へもどったとき，その下の絵は壊れていないはずです．アプリケーションは描きなおしをしていません．Windowsが描き直すのです．そういうところは非常に気まぐれです．

最初は「メニューやパネルは描きなおしにこないんだな」と思いました．ところが，パネルを何枚も開いてだんだんメモリが足りなくなると，ある時点で突然描き直すのをやめます．「ここでパネルを開いてメモリがxxKBあるから大丈夫だな」と思っても，Windowsがメモリをどう使うかはなかなかわからないのです．

マイクロソフトでは，もっと体系だてて考えられているはずですが，プログラムを組むときは「メモリが空いているといってもあてにはならない」と考えたほうがいいでしょう．

アプリケーション作成上の注意

Windowsアプリケーションを作成していく上で，DOSとは大きく異なる点がいくつかあります．これらの点についてはWindows SDKのマニュアルでもあまり触れられていません．

メモリ管理

イメージ処理のように非常に大きなデータを取り扱う場合にメモリ管理の問題がよくわかります．たとえば，スキャナで取り込んだ写真などをファイルから読んで，ディスプレイに表示することを考えてみてください．

Windowsの関数であるCreateBitmap，次にBitBltを実行すると画面にイメージが表示されます．BitBltでは，ビットマップハンドルというものを指定します．ビットマップハンドルはCreateBitmapで作ります．CreateBitmapではデータを読み込んだアドレスを指定します（**図4**）．

バッファにデータを読み込んでCreateBitmapを実行すると，このバッファはもう返却しても大丈夫です．システムのヒープエリアにイメージが転送されて，その先頭がビットマップハンドルという形でユーザーに返ってきます．だから元のバッファは返してもいいのです．こういうところはマニュアルにも詳細には書いてありません．ユーザーから見るとメモリを返却しているのに，システム全体のメモリは変化しないということになります．

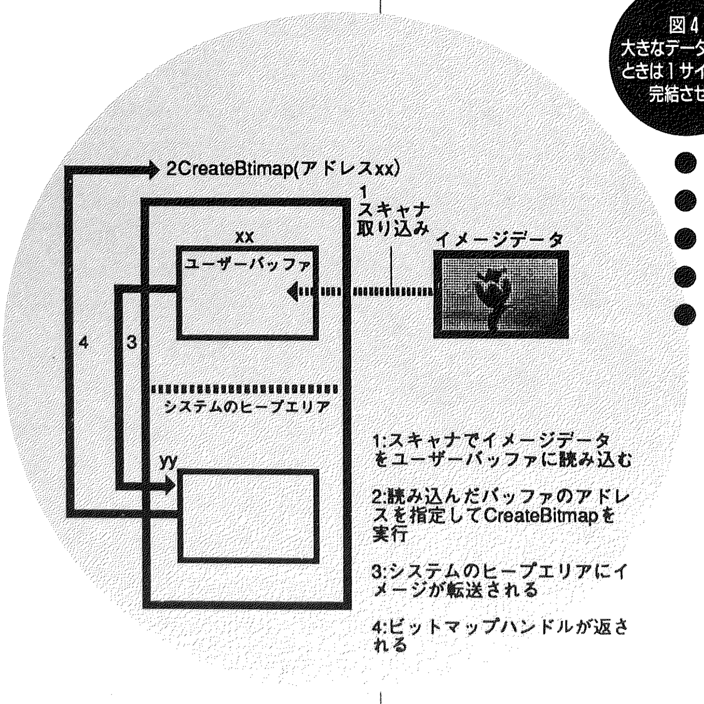
そのうえ一時的には，ユーザーがデータを読み込むために確保したメモリと，CreateBitmap関数によって作られたメモリの両方がシステムに存在することになります．その結果，大きなイメージデータを扱おうとすると一瞬のうちにシステムが死んでしまうことがあります．メモリが取れているように見えても，CreateBitmapを実行した途端に死んでしまうのです．

ビットマップハンドルを作ったらすぐに返さないと，メモリはどんどん足りなくなっていきます．返し忘れたままreturnすると，だいたい死んでしまうというくらいもろいものです．

ファイルアクセス

Windowsでは，ファイル入出力だけは，DOSの標準関数を使うようになっています．ファイルのオープンは，WindowsのOpenFileという関数を

図4
大きなデータを扱うときは1サイクルを完結させる



Windowsプログラミング用語
メタファイル（Metafiles）　GDI関数をバイナリフォーマットにコード化して集めたものをメタファイルという．

使いますが，それ以外のread, seek, writeなどはすべてDOSの標準関数を使えといっています．

DOSの関数を使うと，Windowsの管理からはずれることになります．ファイルをオープンしてからクローズするまでの間にreadやwriteをすると，その間はWindowsに制御がもどりません．ほかのアプリケーションのことを考えたら，その時間はできるだけ短いほうがいいはずです．そこで，通常はオープンからクローズまでを一瞬のうちに終らせて，途中でループさせたりはしません．何か要求が起きると，必要なところだけを読んでディスプレイに表示して終る．open, close, read, writeはこまめにやること．これがひとつの注意事項です．

イメージデータを全部読み込もうとすると，大きなデータバッファが必要なわけです．そこで，プログラマはメモリを分割しようとするでしょう．バッファを確保して，そこへファイルから読み込んでCreateBitmapを実行し，また次を読んできてCreateBitmapを実行するという作業を繰り返せばいいと考えるでしょう．

ところが実際にはビットマップが複数個できて，メモリは足りなくなります．それを防ぐには，1サイクルを完全に終らせなければなりません．バッファを確保して，ファイルを読み込んで，ビットマップを作成して，メモリを返して，ディスプレイに表示してしまうという一連の処理をします．

文字列操作

通常のCプログラムにはcharやshortなどのデータ型がありますが，Windowsには特殊なデータ型があります．頻繁に出てくるものに，LPSTR, LPINTなどがあり，ビットマップならBitmapという構造体があります．RectangleだとアドレスはLPRECTとか書いてあるわけです．実際にはこれらのデータ型はdefine文でchar ＊となっています．DOSの場合と違ってこれらはすべてfarです．

メモリを確保するときにはGlobalAlloc, LocalAllocの2種類があります．GlobalAllocはfar領域にメモリをとります．メモリ領域をとって使うときはたいていfarです．プログラムの入口もfarです．しかし，farでとった領域から別の領域へstrcpyで文字列をコピーできないのです．最初にこれに気づいたときは，ローカルのエリアにバッファをとって，farのところに1文字ずつ移すというやり方でコピーすることにしました．

read, write, それからstrxxxという関数はすべてlong far型を使わなければなりません．ところがlong far型を使わなくてもWindowsは落ちません．ただし結果はメチャメチャになります．その後，文字列コピーをする関数とヘッダファイルを見つけたのですが，その関数は完璧ではなかったため，そういうところは自作したりして補なっています．

Windowsアプリケーションのデバッグ

Windowsではメモリがきついため，デバッガの入る余地がありません．Codeviewというソースレベルのデバッガが提供されていますが，Windows2.xではメモリが足りないので実用的とはいえません．それでは一番簡単なデバッグ方法は何でしょう？

WindowsのSDKを買うとフロッピーの中にNODEBUGとDEBUGという2つのディレクトリがあり，それぞれにWindowsのカーネルが入っています．DEBUGというディレクトリに入っているカーネルをコピーして，それを使ってインストールするとWindowsはDEBUGモードになります．DEBUGモードでWindowsが走るとき，何か異常が発生するとメッセージをRS-232Cポートに出します．RS-232Cポートの先にターミナルをつないでおけば，画面にメッセージが表示されるという仕組みです．

メッセージ自体はほとんど役に立ちません．ただしこのシステムには非常に意味があります．NODEBUGのWindowsでは，プログラムミスで異常が発生したときにはリセットボタンを押さなければなりません．ところがこの形にしておけば，かなりの場合は救われます．リモートターミナルにメッセージが出て，ABORTかRETRYかを聞いてきます．ABORTと答えればDOSのプロンプトにもどります．

私たちはこのメッセージに付加価値をつけています．プログラムの中でRS-232Cに応答型のprintを入れておいて，リモートターミナルに独自のエラーメッセージや確認メッセージを表示させるのです．

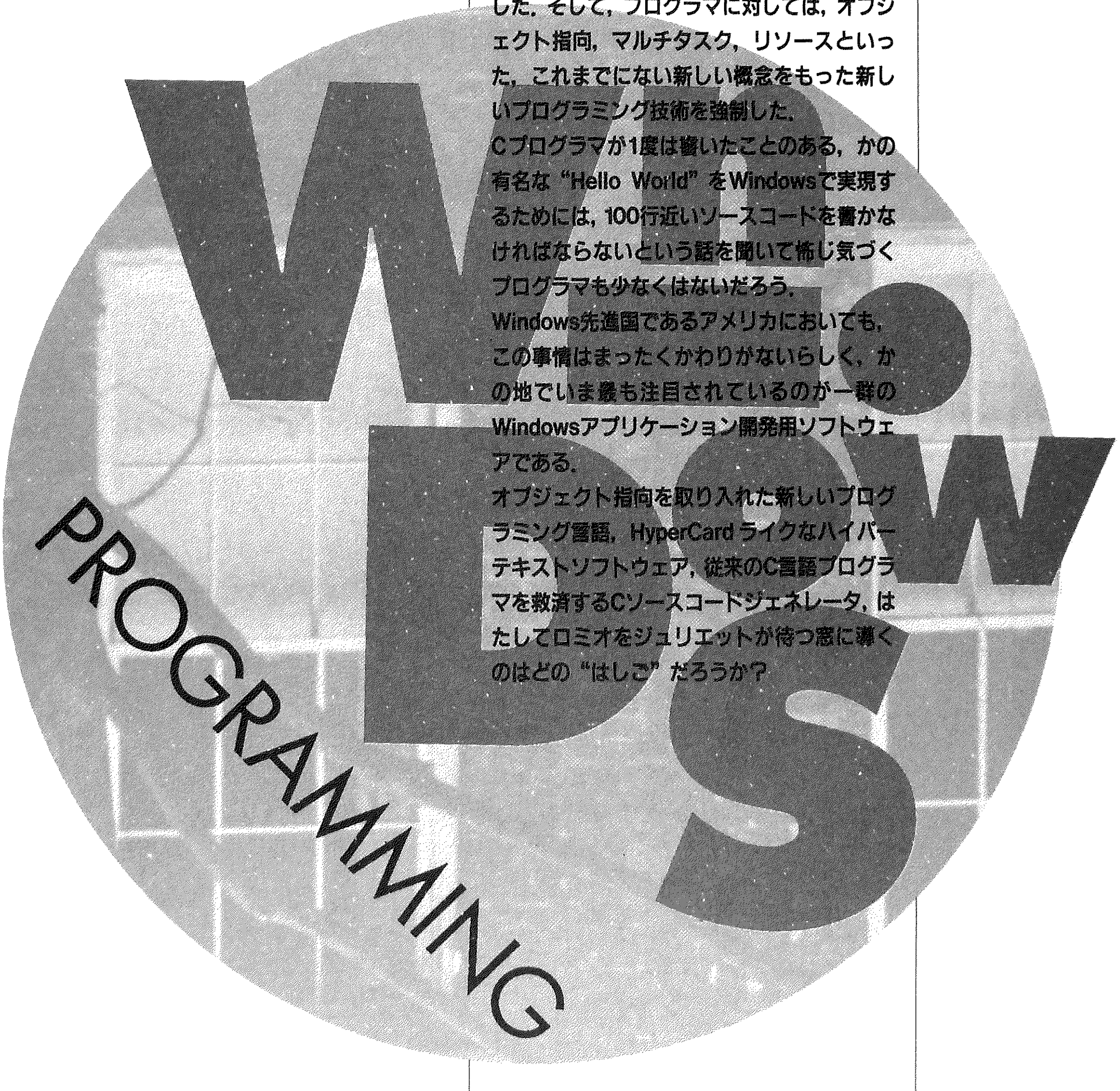
米国版のWindows 3.0でも2.Xと同じデバッグ用のカーネルがありますが，Codeviewが完全に使えるようになっています．PC/ATのスロットにMDA（モノクローム・ディスプレイ・アダプタ）のカードを差して，ディスプレイを付けます．Codeviewのメッセージはすべてここに表示されます．

Windowsプログラミング用語
メッセージ（Messages）　Windowsにおいて Windowsシステムや各タスク間で情報の伝達や，コミュニケーションを行う手段をいう．

**Windows
プログラミング講座
5**

新しいWindowsプログラミング

Windowsは，ユーザーにこれまでのキャラクタベースのMS-DOSの世界では考えられなかったマウスとアイコンによるグラフィカルで直観的なユーザーインターフェースをもたらした．そして，プログラマに対しては，オブジェクト指向，マルチタスク，リソースといった，これまでにない新しい概念をもった新しいプログラミング技術を強制した．
Cプログラマが1度は書いたことのある，かの有名な“Hello World”をWindowsで実現するためには，100行近いソースコードを書かなければならないという話を聞いて怖じ気づくプログラマも少なくはないだろう．
Windows先進国であるアメリカにおいても，この事情はまったくかわりがないらしく，かの地でいま最も注目されているのが一群のWindowsアプリケーション開発用ソフトウェアである．
オブジェクト指向を取り入れた新しいプログラミング言語，HyperCardライクなハイパーテキストソフトウェア，従来のC言語プログラマを救済するCソースコードジェネレータ，はたしてロミオをジュリエットが待つ窓に導くのはどの“はしご”だろうか？



Windows用のアプリケーションジェネレーター WindowsMAKER ver.3.01試用記

西川和久

Windowsプログラマをつねに悩ませるものが，ウィンドウスタイル，ダイアログボックス，プルダウンメニューなどを定義するためのソースの記述である．スケルトンを使用することである程度簡略化することは可能だが，複雑なメニューやダイアログボックスの記述はかなりめんどうな作業だ．これらの作業からプログラマを解放し，さらにWindowsの中からプログラムのコンパイル・リンク・実行までを行ってくれる魔法のツールがWindowsMAKERである．

原始的な開発環境

Windows対応のアプリケーションを作成する場合，Microsoft CおよびMicrosoft Windows Software Development Kit (SDK) を使用するのが普通である．ところが，これらはソフトウェアツールと呼ぶにはあまりにも原始的であり，開発効率は非常に悪い．

Windows 3.0版SDKの内容は，以前の2.11版と比べれば随分改善されたが，まだまだ裸のままであり，必要最低限の機能を満たしているだけといっても過言ではない．これでは，「Windowsのアプリケーションを組むのはたいへんだ！」といわれるのも当然である．このWindowsプログラミングの複雑さからプログラマを解放してくれるツールはないものだろうか？

噂では，マイクロソフト社内に，統合環境＋中間言語からWindows／OS/2 PM／Macintoshへ落としこめるツールがあるという話だが，こんなものが世の中に出てくるわけではない．いっそ切口をかえて，Actorなどに代表されるオブジェクト指向型開発環境に移行するという手もあるが，今までC言語＋SDKでガチガチに組んできたプログラマにとっては，なかなかとっつきにくいしろものである（この辺の関係式をあまりわかっていないプログラマにとっては，非常に有効な開発環境だと思われるが……）．かくいう私もそのようなCプログラマの1人であり，この落差を緩和するよい方法はないものかと思案を続けていた．

SDKでプログラムを組んだことのある人ならば経験ずみのことと思うが，SDKとMS-Cによる開発環境では，Dialog Box Editor, Resource Compiler, SDK Paint, Text Editor, C Compilerの間を行ったりきたりし，Windows上でプログラムの実行・デバッグそしてまたコマンドプロンプトへといった繰り返しがえんえんと続くことになる．このような，統合環境とはあまりにも程遠い環境でアプリケーションを作るのでは，開発効率が上がるはずがない！

そして，このような不自由な環境での開発を続けていると，「Dialog Box Editorである程度の枠組みを作れば，C言語形式のスケルトンくらい自動生成できるのではないだろうか？」という考えが自然と頭の中に浮かんでくるのである．

ここで紹介するWindowsMAKERは，まさにその願いをかなえてくれる統合開発環境なのである．SDKのツール群を有機的に結びつけ，できる範囲でC言語のソースを自動生成し，コンパイル／リンクまでをオートマチックに実行してくれる．つまり，よい意味でMicrosoft C 6.0に付属するPWBのWindows版といえるツールなのである（イメージが悪いかな!?）．

WindowsMAKERのインストール

WindowsMAKERを使うにあたって最低限必要なものは，

1. Windows 3.0
2. Microsoft C Compiler (5.1または6.0)
3. Windows Software Development Kit (SDK) Ver.3

の3つである．WindowsMAKERは，インストールの際にこれらのディレクトリを調べてチェックしている．

Windowsプログラミング用語
ラジオボタン (Radio botton)　ダイアログボックス内などで複数の選択肢から排他的な選択をするボタン．オプションボタンともいう．

これらのソフトが1つでも欠けていると，インストールプログラムはエラーメッセージを表示し，何もインストールせずに終了してしまう．

正常にインストールが終了すると，いったんWindowsから抜けるよう指示があるのでWindowsを終了し，再起動すると，タイトルバーの左端にあるコントロールメニューボックスの隣にキャンドルマークが表示される．これをクリックすると写真1のようなメニューが表示されるので，この中のWindowsMAKERを選択すると，めでたく本体が起動する．

Windowsプログラミングの複雑さを解消

WindowsMAKERの使用法は，SDKに付属するDialog Box Editorを使ったことのある人には簡単に理解できるだろう．

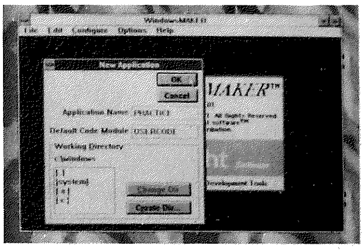
WindowsMAKERの中から，パーツの選択・設定，表示データの入力，Compiler・Linkerの設定などを自由に行うことができ，Windowsプログラミングに必要な作業のほとんどをマウスを使った簡単な操作で行えるようになる．実際の開発の様子については，次章をご覧いただきたい．

SDKのカラクリをすでに理解している者にとっては，WindowsMAKERがShellの役目を果たし，開発効率を大幅に上げてくれる．

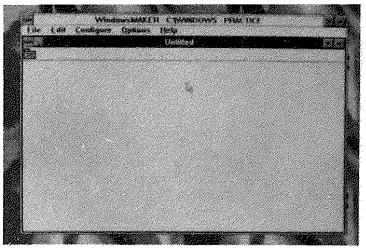
また，WindowsMAKERを使用することで，Windowsプログラミングを難解なものにしている多くの部分が覆い隠され，直接プログラマがコーディングする必要がなくなるため，これからWindowsプログラミングに挑戦しようとしているプログラマにとってはまたとない有効な開発ツールとなるはずである．

WindowsMAKER　595ドル
発売元：Candlelight Software
2375 East Tropicana Ave.,
Suite 320
Las Vegas, Nevada 89119 USA
Tel. (702) 456-6365
Fax. (702) 434-0580
＊価格は米国内での販売価格です．

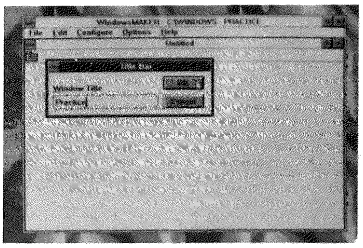
WindowsMAKERによるアプリケーション開発



1．まずWindowsMAKERを起動し，FileメニューからNewを選択してワーキングディレクトリとこれから作成するプログラムの名前（ここではPractice）を入力する．



2．OKボタンをクリックすると，これから作成するウィンドウが表示される．



3．EditメニューからTitle Bar...を選択し，ポップアップしたウインドウ内で作成しているウィンドウのタイトル名を入力する．

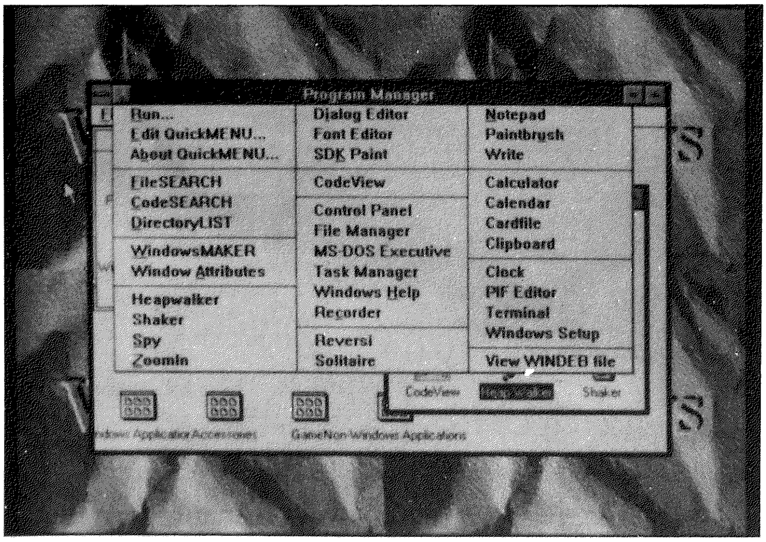
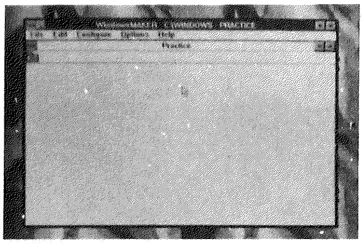
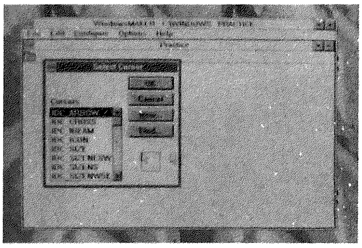


写真1　WindowsMAKERのQuickMENUを表示したところ

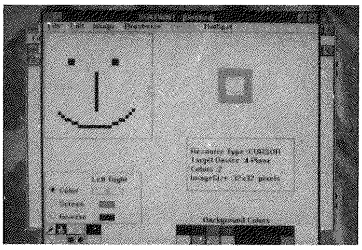
Windowsプログラミング用語
ラスタオペレーション（ROP）　グラフィックイメージを描画する場合，ピクセル間での論理的な操作をラスタオペレーションという．



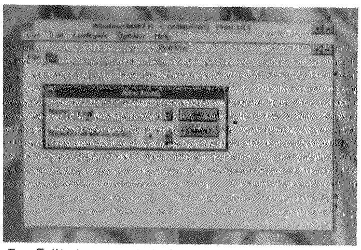
4. 表示されているウィンドウにタイトル名が入る.



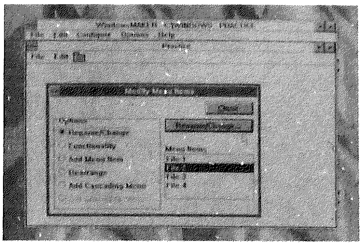
5. Editメニューから Cursor...を選択すると SelectCursorウィンドウがポップアップする. ここで既存のカーソルを選択するか, または新しいカーソルを作成する.



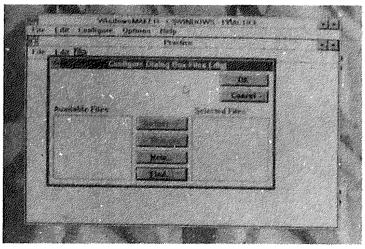
6. Newボタンをクリックすると自動的にSDK Paintが起動し, 新しいカーソルを作成することができる. アイコンも同様にして作成できる.



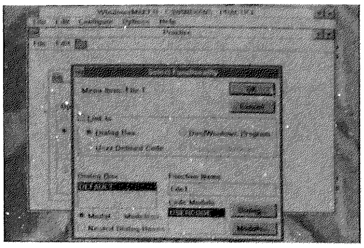
7. Editメニューから Add Menu...を選択することで, メニューバーにメニューを作成することができる. ここではFileとEditを作成している.



8. メニューバーに作成されたメニューをマウスでクリックすると, Modify Menu Itemsウィンドウがポップアップする. ここで, そのメニューの内容を定義する. メニューの追加, 削除, 並べ替えなども行える.

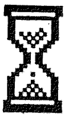


9. ConfigureメニューからDialog Box Files (.dlg)...を選択すると, Configure Dialog Box Files (.dlg) ウィンドウがポップアップする. ここで, 既存のダイアログボックスを自分のプログラムに付加したり, 新たにダイアログボックスを作成したりすることができる.



10. ダイアログボックスをメニューにわりあてるには, そのメニューを選択してModify Menu Itemsウィンドウのオプションで Functionalityを選び, さらにポップアップしたSelect FunctionalityウィンドウでLink toをクリックし, わりあてたいダイアログボックスの定義ファイルを選べばよい.

11. すべての定義が終了したならば, Fileメニューから Save and Generate Codeを選択する. すると必要なソースファイルそのほかが作成される (CHART). 次にMake Applicationを選択すると, NMAKEまたはMAKEが起動され, プログラムの実行形が作成される. Run Applicationを選択すれば, 作成された実行形の動作を確認することができる.



CHART

WindowsMAKERが作成するファイル（プログラム名がPracticeの場合）

PRACTICE	NMAKEまたはMAKE用のスクリプトファイル
PRACTICE.C	Cのソースファイル
PRACTICE.DEF	モジュール定義ファイル
PRACTICE.H	ヘッダーファイル
PRACTICE.LNK	リンクインプットファイル
PRACTICE.RC	リソースファイル
USERCODE.C	ユーザー定義のCソースファイル

Windowsプログラミング用語
ラスタフォント（Raster fonts）　特定のサイズでデザインされ, ドットで構成されている. サイズを自由に変更できない.

Windows版 ハイパーテキストソフトウェア Asymetrix ToolBook

1990年5月22日に米国で発表されたWindows 3.0には，Asymetrix社のハイパーテキストソフトウェアであるToolBookのランタイム版が添付されていた．これは，あたかもMicrosoft社がWindows上のソフトウェアプラットホームとしてToolBookを認めたかのような印象をユーザーに与えた．はたしてToolBookは，MacintoshにおけるHyperCardのようにWindowsのソフトウェアプラットホームになりえるのだろうか？

Software Construction Set

ToolBookは，プログラマではないユーザーが自分自身のためにソフトウェアを開発するための環境を与えてくれるソフトウェアである．また同時に，そのようにして作られたソフトウェアを実行する環境をも与えてくれる．

Windowsのような複雑なシステムの上で動作するアプリケーションを開発しようとすると，プログラマはそのシステムに関する知識を得るのにかなりの労力と時間を割かなければいけなくなる．実際，Windows 3.0のAPI (Application Programming Interface) は約600にものぼり，これらを理解してプログラムを組むのはそう容易なことではない．

ToolBookは，これらの複雑さをユーザー（プログラマ）の目から覆い隠し，自然言語（英語）に近い言語によるスクリプトの記述とグラフィカルな操作によって，容易にアプリケーションを作成することを可能にしている．

Macintosh上のHyperCardと同じく，ToolBookはWindows上のアプリケーションプラットホームとなる可能性を秘めたSoftware Construction Setというべきソフトウェアなのである．

ToolBookの構造

ToolBookでは，情報のひとかたまりをページという概念で表し，このページが集まったものをブックと呼んでいる．ページは1つのバックグラウンドと1つのフォアグラウンドからなり，1つのブックは複数のバックグラウンドをもつことが可能である（図1）．

ページは，テキスト／イメージ／テキスト入力が可能なエリアなどを含むことができる．

バックグラウンドは，フィールド／グラフィック／ボタン／レコードフィールドの4つを含むことができ，フォアグラウンドは，フィールド／グラフィック／ボタンの3つを含むことが可能である．

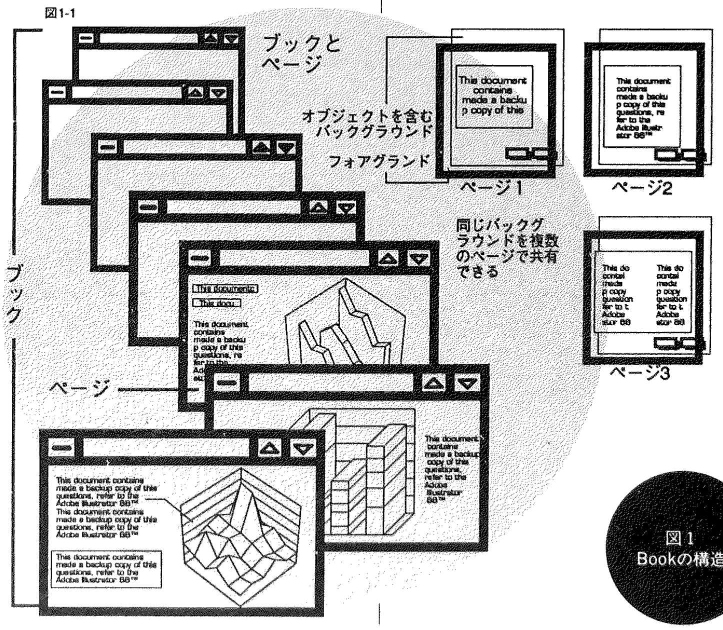


図1 Bookの構造

Windowsプログラミング用語
ランドスケープモード（Landscape mode）　プリンタにおいて長辺を上にして用紙を入れるモード．短辺を上にして用紙を入れるのはポートレイトモード．

ToolBookはオブジェクト指向の開発システムであるから，ブック，ページ，バックグラウンド，ページ上のあらゆる要素などはすべてオブジェクトとして認識される．マウスのクリックのようなイベントが生じると，ToolBook内でメッセージが送られ，このメッセージを受け取ったオブジェクトは前もって決められた動作を行う．このように，イベントによって生じたメッセージをオブジェクト間でやりとりすることでToolBookのアプリケーションは動作するのである．

また，ToolBookにはReader levelとAuthor levelの2つのユーザーレベルが存在する．ユーザーは，Reader levelでToolBook用に作られたアプリケーションを使用し，Author levelでToolBook用のアプリケーションを作成することになる．

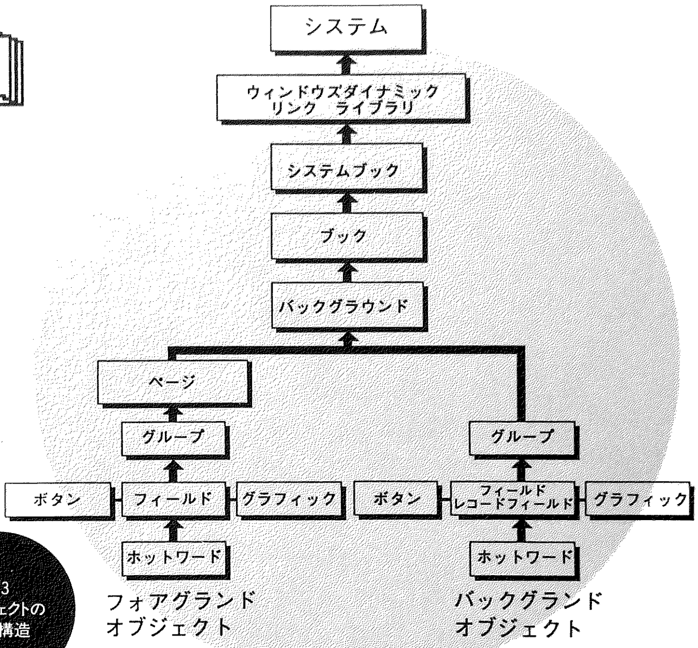


図3 オブジェクトの階層構造

OpenScript

ToolBookにはOpenScriptと呼ばれるオブジェクト指向言語が備えられており，この言語を用いて新たなアプリケーションを作成することができる．OpenScriptで書かれたプログラムをスクリプトと呼んでいる．

ToolBookにおけるプログラミングとは，オブジェクトを作り，そのオブジェクトの動作を定義するスクリプトを書くことである．スクリプトを記述することで，オブジェクトの動作を変更したり新しい動作を付け加えたりすることが可能になる．

スクリプト中のそれぞれの処理はステートメントと呼ばれ，ステートメントが集まったものがハンドラーと呼ばれる（**図2**）．オブジェクトがメッセージを受け取ると，ToolBookは，そのオブジェクトのスクリプト中にそのメッセージに対応するハンドラーがあるかどうかを見て，もし存在すればそのハンドラー中のステートメントを実行する．

対応するハンドラーが見つからないときには，それが見つかるまで，メッセージはオブジェクトの階層を上に向かって送られていくことになる．オブジェクトの階層構造は図3に示す．

実際にスクリプトを記述するには，2つの方法がある．1つはスクリプトウィンドウを開いて直接キーボードから入力する方法であり，もう1つはスクリプトレコーダーを用いて動作（キーストロークおよびマウスアクションの結果）を記録する方法である．

ToolBookには，スクリプトレコーダー以外にもプログラミング支援ツールとして，OpenScriptのコマンドを直接実行できるコマンドウィンドウやスクリプトのステップ実行が可能なデバッグウィンドウなどがある．

また，Windowsのダイナミックリンクライブラリの機能を用いて，C言語，Pascal，アセンブラなどでOpen-

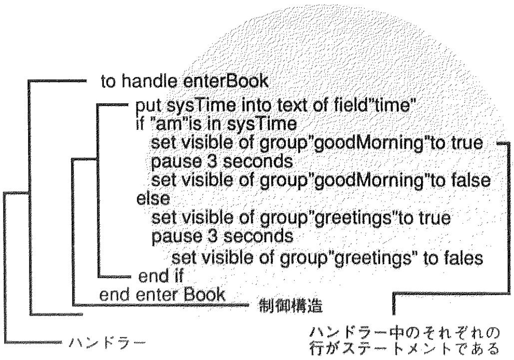


図2 スクリプトの基本構造

Windowsプログラミング用語
リソース（Resource） 各Windowsプログラムで使われるデータやオブジェクトのこと．ソースから独立することで共有化をはかっている．

Scriptの機能を拡張できる．

アプリケーションの作成

では，実際にToolBookのアプリケーションを作ってみたい．ここでは，簡単なフロッピーディスク管理用のデータベースを作ってみる．

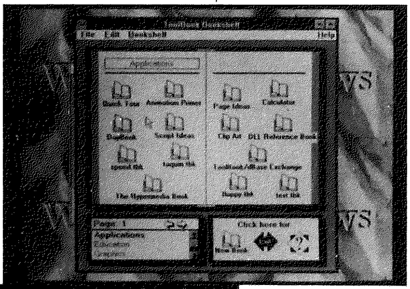
まずToolBookを起動すると，Bookshelfと呼ばれる画面が現れる（**写真1**）．ここで，Bookshelfの右下にあるNew Bookをマウスでクリックして新しいBookを開く（**写真2**）．次に，EditメニューからAuthorを選んでAuthor levelに入る．すると，画面にツールパレットが表示され，Object，Draw，Windowのメニューが現れる（**写真3**）．

Author levelに入ったなら，PageメニューからBackgroundを選んでBackgroundに移る．ここでツールパレットを利用して，フィールド，ボタン，グラフィックを作成する（**写真4**）．ツールパレットは，Autohr levelにいるときはいつでも利用できる．

作成したレコードフィールドの細かい設定は，ObjectメニューからRecordField Properties...を選び，ポップアップしたウィンドウ中で行う（**写真5**）．さらにScript...ボタンをクリックすればスクリプトウィンドウが開き，OpenScriptのスクリプトを記述することができる．また，Textメニューを使って，レコードフィールドに表示する文字のフォントや大きさを決めることも可能である．ここでは，たんなる四角とスクロールバーつきの2つのレコードフィールドを作成した．

次に，このBookを終了してBookshelfに帰るボタン，前ページに戻るボタン，次ページに進むボタンの3つを作成する．まずボタンのグラフィックを作成し，それぞれのボタンの動作をスクリプトウィンドウを開いて記述する（**写真6**）．記述が終わったところで，ScriptメニューからCheck Syntaxを選んでスクリプトの文法チェックを行う．記述に誤りがなければSyntax OK，誤りがあればSyntax errorというウィンドウがポップアップする（**写真7**）．もしerrorがでたならば，ScriptメニューからDebugを選んでデバッグウィンドウを起動し，デバッグを行う（**写真8**）．最後に，必要なタイトル文字を記入してフロッピーデー

写真1　Bookshelf



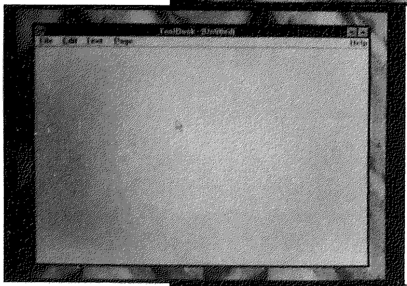


写真2
新しいBookが起動する

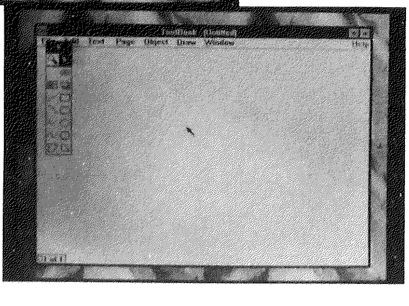


写真3
Author levelに入るとツールパレットが表示される

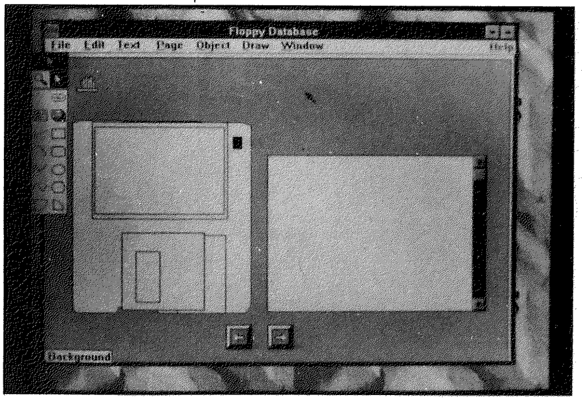


写真4　レコードフィールド，ボタン，グラフィックを作成

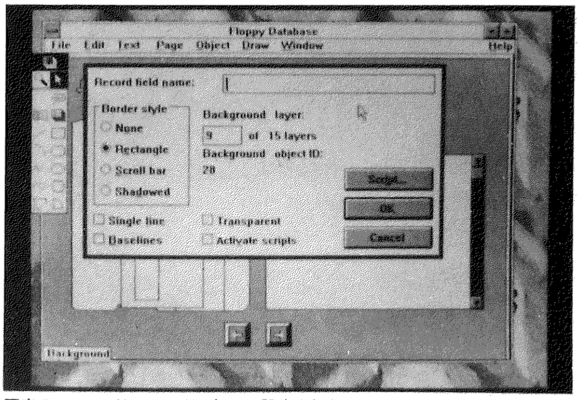


写真5　レコードフィールドの細かい設定を行う

タベースは完成である（**写真9**）.

ここでは使用しなかったが，ToolBookでは適当な文字列をホットワードとしてボタンと同じように使用することができる．また，複数のオブジェクトをひとつのグループとしてまとめて，あたかもひとつのオブジェクトであるかのように扱うことができる．これは，アニメーションなどを作成するときにとくに有効である．

コピー&ペーストで簡単プロトタイピング

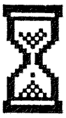
ToolBookには，Page Ideas, Script Ideas, Clip ArtといったサンプルやプログラミングのためのBookが豊富に付属している．このため，ユーザーは自分自身で1行もスクリプトを記述することなしに，これらのBookに含まれているオブジェクトやスクリプトをコピーして自分のBookにペーストするだけで，いとも簡単にアプリケーションを作成することができる．

コピー&ペーストでアプリケーションを作成しながら，必要になったところでOpenScriptの学習をするというのが，最もうまいToolBookの利用法かもしれない．

このようにじつに簡単にアプリケーションの作成ができるToolBookだが，大きな欠点が1つある．それは，その動作スピードがあまりにも遅いという点だ．はっきりいって，メインメモリ2MBの386マシンではとても使う気にならない．

ソフトウェアとしてのできはMacintoshのHyperCardにすぐるとも劣らないだけに，この点だけはなんとか改良してほしいものである．

ToolBook Ver.1.0　395ドル
発売元：Asymetrix Corporation
110 110TH Avenue N.E.,Suite 717
Bellevue,WA 98004
Tel.(800) 624-8999
Fax.(206) 454-0672
＊価格は米国内での販売価格です．
（日本語版はジィスクソフトウェア（株）から発売予定）



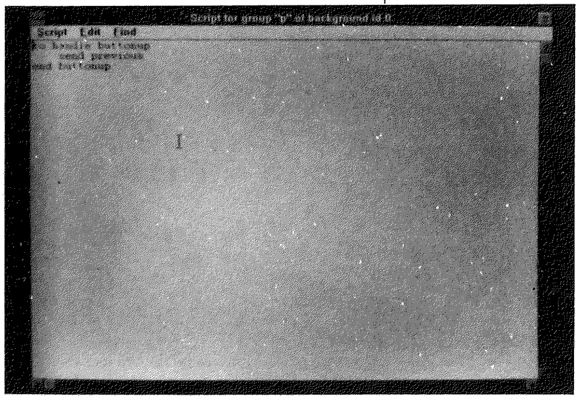


写真6　スクリプトウインドウ

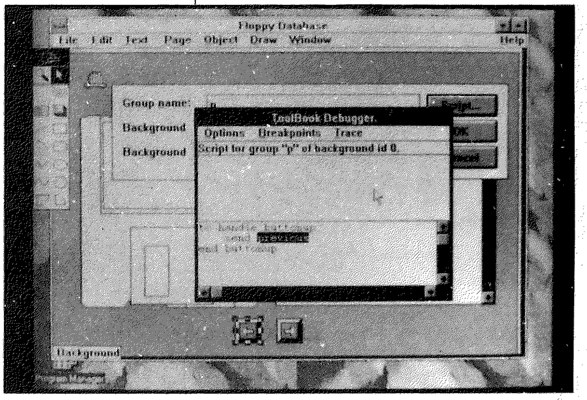


写真8　デバッグウインドウ

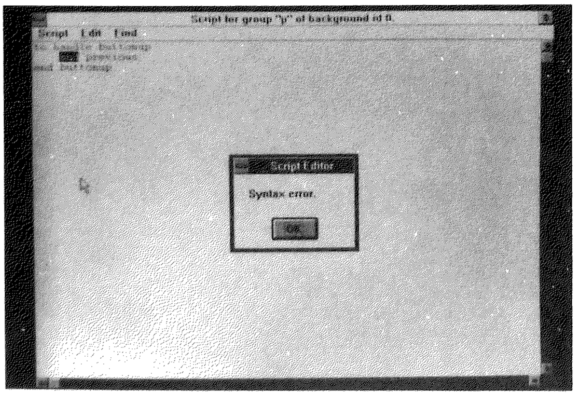


写真7　スクリプトの文法チェックを行う

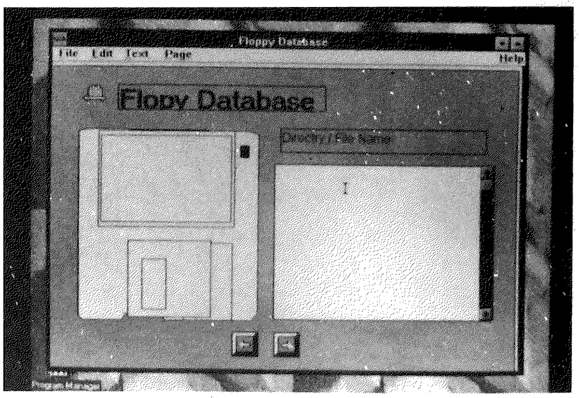


写真9　完成したフロッピーデータベース

Windowsプログラミング用語
リソースファイル（Resource files）　アイコンやメニューなどの定義されているデータの集まり．プログラムは必要に応じてこのファイルを利用する．

Windowsのための オブジェクト指向開発システム Actor 3.0

Actor 3.0は，米国The Whitewater Groupが開発，販売しているWindowsアプリケーション開発システムである．米国における評価は高く，PC MAGAZINE誌で表彰もされている．Windows用に特殊化されたその開発環境についてレポートする．

オブジェクト指向言語Actor

オブジェクト指向言語について，すでにご存知の方も多いと思うが，簡単に説明しておきたい．

いま流行しているC言語などは手続き型言語と呼ばれ，プログラマはデータ構造を設計し，このデータを扱う手続きや関数を作成することでプログラミングを行う．これに対してオブジェクト指向言語では，データとそのデータに施す操作（メソッドと呼ばれる）をまとめてオブジェクトという単位を構成する．オブジェクトになにかをさせるには，そのオブジェクトにメッセージを送るということを行うのである．メッセージを受け取ったオブジェクトは，そのメッセージに対応したメソッドを実行する．

つまり，オブジェクト指向言語で作成されたプログラムは多数のオブジェクトの集合体であり，それらのオブジェクト間でメッセージをやりとりすることでプログラムが動作することになる．

現在，オブジェクト指向言語として，C言語のスーパーセットであるC++やPascalを拡張したオブジェクティブPascalなどが存在しているが，Actorはこれらのオブジェクト指向言語とは一線を画すSmallTalkと同様，純粋なオブジェクト指向言語である．

継承とクラスライブラリ

オブジェクト指向言語におけるオブジェクトは，クラスごとに階層化されて管理されており，プログラマは既存のクラスを元にして新たなクラスを定義することができる．このとき，新たに作られたクラスは元のクラスの性質を継承している．

この継承の機能は非常に重要である．親のクラスがもっている機能を新しいクラスで実現するために，プログラマは1行のコードも書く必要はない．継承によって，プログラマは既存のクラスを元に，より高機能な新しいクラスを容易に作成することができるのである．結果的にプログラマが保守すべきコードの量は減り，製品の開発期間は大幅に短縮される．

さて，ここで問題になるのがクラスライブラリである．せっかく継承という機能があっても，プログラマが1からクラスを作っていかなければいけないとしたら，これはたいへんなことである．そんなことをするくらいなら，手続き型言語を利用してプログラミングしていたほうがましだということになっても不思議はない．実際，C++などのオブジェクト指向言語がいまひとつ普及しないのは，そのクラスライブラリの不備に原因があるといわれているくらいである．

しかし，Actorには購入時から豊富なクラスライブラリが提供されているので，このような心配はない．

Actorと既存言語

Actorは，Windowsのダイナミックリンクライブラリのルーチンをコールできるため，Actor以外の言語で作成されたコードを利用することができる．このため，WindowsやActorが提供していない機能も，市販されているC言語のライブラリ製品などを利用することで簡単に実現することが可能となる．

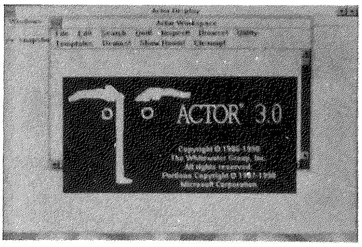


写真1　Actorの起動画面

Windowsプログラミング用語
リロケータブルオブジェクト（Relocatable objects）　メモリマネージャによってメモリ上のどこへでもロードすることのできるオブジェクトのこと．

Windows環境で開発できるActor

マイクロソフト社の提供するWindows開発環境であるWindows Software Development Kit(SDK)とMS-Cの組み合せでは，プログラムを開発している途中でなんどもDOSとWindowsの間をいったりきたりしなければならない．これは，Windowsアプリケーションを開発するプログラマに大きな心理的ストレスを与えるばかりでなく，貴重な開発時間の浪費を引き起こす元凶でもある．

Actorは，それ自身がWindowsアプリケーションである．このため，アプリケーション開発時にWindowsを出る必要がなく，Windows上でプログラムのコーディング，実行，デバッグが行える．

ただし，最終的なアプリケーションプログラムの実行形を作るときには，Windowsを出てリソースコンパイラを使用する必要がある．

プログラム開発の実際

C言語などでは，一般にエディット，コンパイル，リンク，プログラムの実行，デバッグという流れで開発が行われるが，Actorでは，クラスライブラリのブラウジング，新しいクラスの作成，メソッドの記述，プログラムの実行，デバッグ，Seal-Off操作，リソースコンパイラによる最終的な実行形の作成という流れでプログラム開発が行われる．

これから開発するアプリケーションをどのようなオブジェクトに分割するかきまったら，Actorを起動しBrowserを呼び出す．Browserを用いてクラスライブラリから必要なオブジェクトを探し出し，これを元に新しいクラスを作成し，メソッドを書き加えていく．1つメソッドを記述するたびにActorはメソッドをコンパイルしてクラスに付加していくのだが，同時にActorは文法チェックを行っており，単純なタイプミスなどはすべてここでクリアされる．

メソッドを記述し終えたら，Workspace上で今書いたプログラムを実行させてテストを行う．もし問題があればエラーメッセージが表示されるので，必要ならばデバッガを起動し，コードを修正する．Actorのデバッガでは，オブジェクト間のメッセージやメッセージ中で使われている変数などを

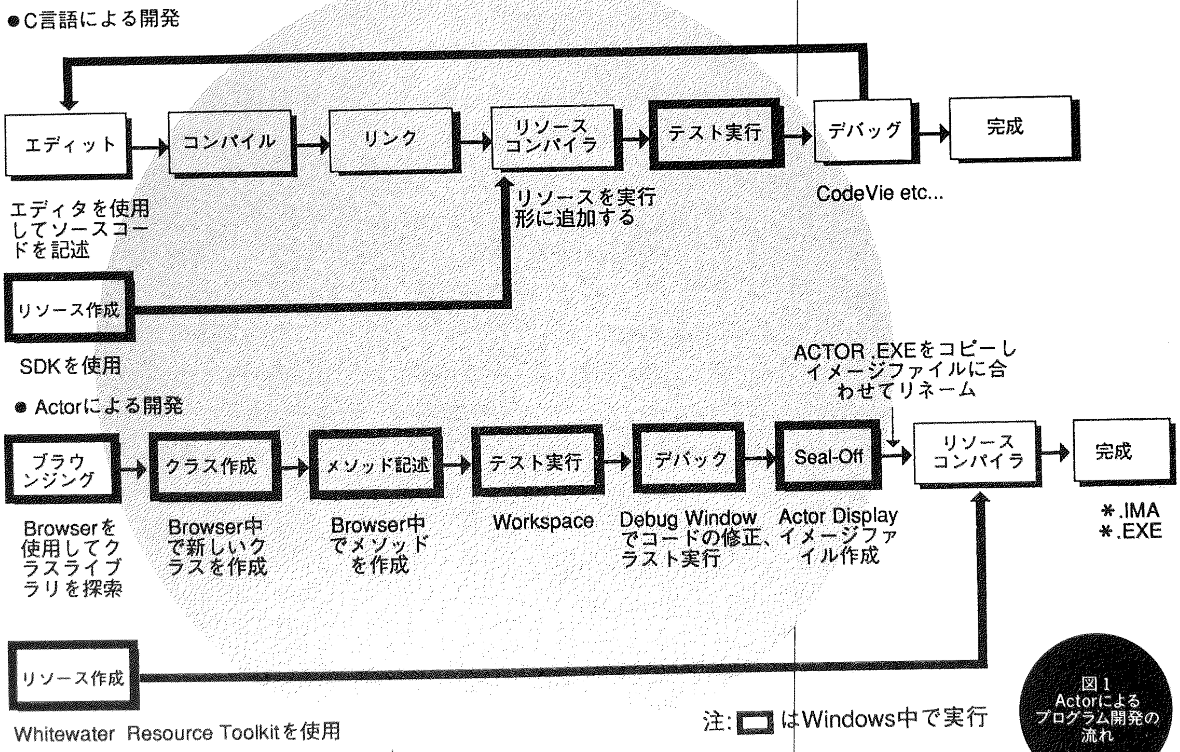


図1 Actorによるプログラム開発の流れ

Windowsプログラミング用語
ローカルメモリ（Local memory）　Windowsのプログラムにおいて，DSおよびSSレジスタはローカルメモリを指している．

調べることが可能である.

すべてのオブジェクトをWorkspaceでチェックしたならば，Seal-Offと呼ばれる操作を行う.

Actorは，コンパイル後のコードをイメージファイルとしてセーブしているが，この中にはActorのツールや余分なクラスが含まれている．これらを切り離して，アプリケーションに必要なコードのみのイメージファイルを作成するのが，Seal-Offである.

次にACTOR.EXEをリソースコンパイラの存在するディレクトリにコピーし，Seal-Offされたイメージファイルと同じ名前にリネーム（ただし，拡張子は.EXEとする）し，リソースコンパイラを用いてリソースファイルを付加する.

こうして作られた.EXEファイルとイメージファイルの2つが，Actorによって作成されたアプリケーションということになる．この2つのファイルを同じディレクトリにおくことで，独立したWindowsアプリケーションとして実行できるわけである.

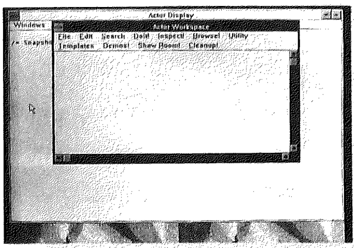


写真2　Workspace

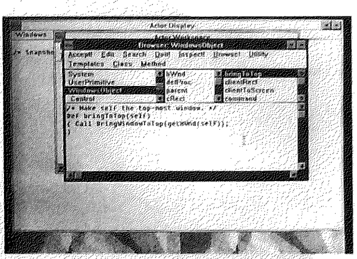


写真3　Browser

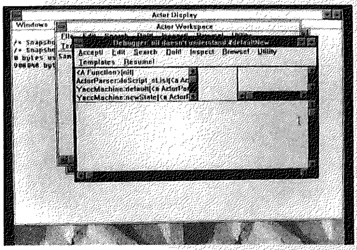


写真4　Debugger

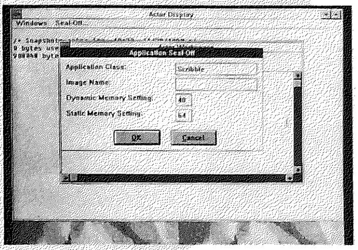


写真5　Seal-Off

Actorの支援ツール

Windowsアプリケーションを C言語を用いて作成する場合，Windows Software Development Kit(SDK)が不可欠だが，Actorにはリソースコンパイラと基本的なリソースファイルが付属しているので，SDKは不要である．ただし，SDK PaintやDialog Editorが付属していないので，オリジナルのリソースの作成はかなりめんどうなものになる.

そこでWhitewater Groupでは，ダイアログボックス，ビットマップ，アイコン，カーソルなどのリソースを作成するためのツールをWhitewater Resource Toolkit(WRT)として発売している．WRTは，あらゆるリソースを対話的に画面上で確認しながら作成することを可能にしてくれる．また，WRTを使用すると.EXE中のリソースの編集も行える.

WRTで作成されたリソースは，C言語で作成したプログラムからも利用可能である.

WRT以外にも，ISAMファイルマネージャであるWinTrieveやグラフィックスライブラリであるObjectGraphicsなども発売されている.

プロトタイピングに有効

Actorは，MS-Cなどのコンパイラとは異なり，本質的にはインタープリタである．このため，Actorで開発したアプリケーションを実行するためには必ずACTOR.EXE（実際にはリネームされたもの）が必要になる．結果的にActorで開発されたアプリケーションの実行形は，C言語で開発されたものよりもかなり大きなものとなってしまう.

しかし，このような欠点はあるものの，その開発環境の使いよさとプログラミングの容易さはMS-C+SDKの比ではない.

C言語のプロにとっても，プロトタイピングツールとして有効に利用できることだろう.

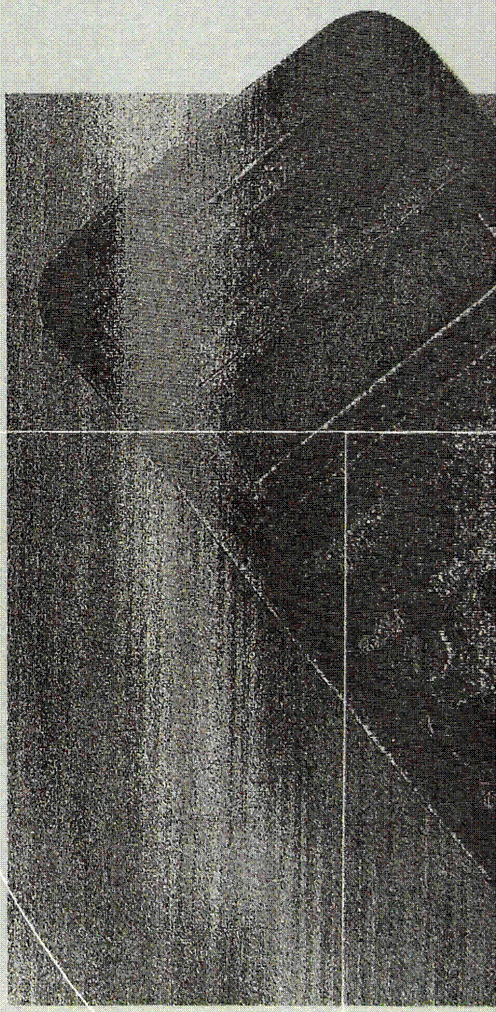
Actor 3.0　695ドル
Whitewatre Resource Toolkit　195ドル
WinTrieve　395ドル
ObjectGraphics　395ドル
発売元：The Whitewater Group
1800 Ridge Avenue Evanston, IL 60201-3621
Tel. (800) 869-1144
Fax. (708) 328-9386
＊価格は米国内での販売価格です.
（総販売代理店　（有）データポット 052-798-4154）

Windowsプログラミング用語
論理座標（Logical coordinate）　Windowsでは直接座標の指定を禁止している．すべての座標は論理座標を設定しGDIによってクライアントエリアに出力される.

祐安重夫

プログラマの道具学

第2回 電卓



プログラミングをするときに，どうしても手元にないとこまるもののひとつは，電卓である．目の前にパーソナル・コンピューター—電子計算機がおいてあるにもかかわらず，別に電卓——電子式卓上計算機が必要というのは，一見奇妙におもえるかもしれないが，やはり必需品なのである．

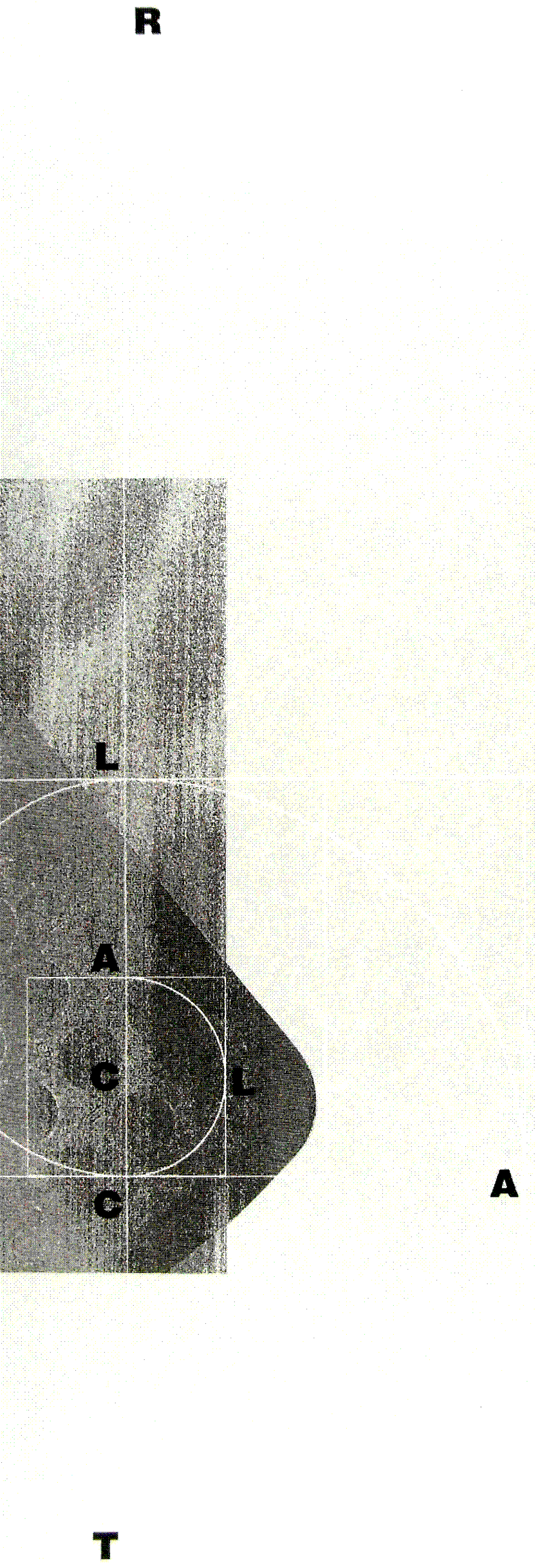
最近のパーソナル・コンピュータやワークステーションでは，マルチ・ウィンドウ・システムなどを利用して，画面に表示された電卓のパネルを，キーボードやマウスで操作することで，計算がおこなわれるようになっているものがある．そこまでいかなくても，たとえばUnixのbcやdcのように，キーボードで操作する簡単な電卓のシミュレータがつかえる場合もある・

それでもやはり，おおくの場合は本物の電卓に手がいってしまう．マウスの操作性は，指で電卓のキーをおすことにくらべるとはるかにおとるし，コンピュータのキーボードよりは電卓のキーの方が，機能性を重視しているのであつかいやすい．

作家の小松左京が『日本沈没』をかいているとき，どうしても大量の複雑な計算をする必要があって，当時はまだ数十万円という価格のつけられていた電卓を購入したという話をきいたことがある．その後電卓の価格はさがる一方で，現在では千円以下のものさえある．

十数年前にはじめて関数電卓を購入したときは，数万円の値段がついており，かなり店頭でまよったものだった．現在ではそれにプログラミングの機能がついたものでさえ，10分の1以下の価格で手にいれることができる．

現在つかっている電卓は米国のHewlett-Packard社のHP-16Cといい，Computer Scientist用と明記されたものである．7年ほど前に数万円で購入したものだが，この製品はなぜか今では製造されておらず，すこし機能のことなるものが，おなじような値段で販売されている．どちらにしても，現在の電卓の価格水準からすれば，かなり高価なものといえる．



この電卓は10進数だけではなく，２進数や８進数，16進数といったものが使用でき，各種の論理演算やシフト演算などがおこなえるので，プログラムのデバッグ時にはかなり役にたつ.実はプログラム電卓でもあるのだが，その機能はつかったことがない．プログラムが必要なら，コンピュータ上でおこなう方がはるかに簡単である．

Macintoshなどでは，この電卓と同等の機能をもったデスク・アクセサリ（アプリケーション実行中にわりこんで動作する，時計や電卓，メモ帳などのプログラム)がつくられ，パブリック·ドメインになっているようだが，やはり本物にはなかなかおよばない．

この電卓にかぎらず，Hewlett-Packard社の電卓のほとんどは，普通の電卓とはちがって，逆ポーランド記法という入力方法を採用している．普通の電卓なら，

15＋32＝

などと入力するところを，

15［enter］32＋

のようにするのである.［enter］は表示されているデータをスタックにいれろという意味であり,＋や－などはスタックの先頭のデータと，現在表示されているデータとの間で，演算をおこなうことを指示するものなのである．

日本語であらわせば，「15たす32は」ではなく，「15に32をたすと」といった具合になり，より自然な日本語の表現にちかいというかんがえかたもなりたつ．またこのような計算方式は，プログラミング言語をコンピュータ内部で処理する際にもよくつかわれており，プログラマにとってはなじみやすいものでもある．

Illustration：MIKIO SASAKI

オダジマタカシのただいま悪戦苦闘中

十年大昔，あるいは顧みられることのない腐敗する無機物の記憶

小田嶋隆

「十年ひと昔」というのは，ひと昔前のことわざであって，少なくともパソコン業界にあてはまるものではない．我々，世紀末のオールドボーイズは，むしろ

「十年大昔」

というふうに，この世界を見るべきだ．

十年たてば，あらゆるものは風化し，崩壊し，壊滅し，無意味になり，カビ臭くなり，腐敗し，灰になる．そして，埋葬され，荼毘にふされ，焼き尽くされ，忘れ去られ，最終的には不潔な思い出になる．例外はない．

思い出は古びることがない，なんてことを誰か（たとえば村上春樹あたり）が言ったことがあるかも知れない．が，その野郎は何も知らないのだ．思い出は，思い出になった瞬間に既に腐っている．なぜなら，そもそも思い出とは，腐敗した記憶を指す言葉なのだから．

とりわけ，マシンにまつわる思い出は，時がたてばたつほど，サビたような，嫌な匂いを立てるようになる．

砂山の砂を
指で掘ってたら
真っ赤にサビた
マックプラスが出てきたよ

ご覧の通りだ．マシンの思い出は，間抜け過ぎて，歌にさえならない．

無機物は腐らない

と教えた化学の教師はとてつもない嘘つきだった．この世の中に存在するすべてのものは，腐る運命のもとにある．概念や構想やプログラムのようなものでさえ，条件さえ整えば，ものの見事に腐る．事実，N-88BASICは生まれたその時に既に腐っていたし，私の知っているマシンたちは，生きてもいないのに死ぬことができた．

思い出してみよう．PC-100はどこへ行ったのか．華々しくデビューした3インチディスクはどうして消え，いかなる理由でOS9は前線を退き，if800は悪い冗談になったのか．そして，この三千世界の一体誰がスーパーMZの墓を守り，FM8の骨を拾い，

六本木パソコンの霊

をなぐさめることができるのか．

謎だ．

すべては，謎だ．

しかも，これらの謎には何の神秘もない．解くにさえ値しない，二束三文の謎，だ．

しかし，ともかく，この十年を振り返ってみることにしよう．誰だって，たまには振り返ってみなければならない．一体に，この業界の人間は，前を見るばかりで，あまりにも，顧みることをしない．これは良くないことだ．孔子のような賢人でさえ，日に三回はわが身を顧みたというではないか．

パソコン世界の10年前は，地球の歴史になぞらえていえば，先カンブリア代に相当する．

10年前，パソコンの世界には，まだ草木の1本どころか，単細胞生物すら存在していなかった．地上の上には，ただ空があるばかりで，その，ただ広いだけの空と荒涼とした地面の間には，酸素さえろくに含まれていない太古の空気が意味もなく循環していた．そして，陸地は移動と衝突を繰り返し，大陸はまだ形成されておらず，ふるちんのアダムと丸出しのイヴは，まだアップルの実の味を知らなかった．

私がパソコンと関わりを持つようになったのは，約7年前，つまり，古生代三畳紀の頃だ．当時，我々の住んでいた海底には

8ビットの三葉虫

とアンモナイトみたいなデータレコーダーがうごめきまわり，顎のない魚が，ようやく泳ぎ始めていた．

私は，その古生代のうす暗い海底に座り込んで，鰓呼吸をしながらパソピア７というマシンの入門書（『親子で楽しむパソピア７』という本であった）を書いた．さらにそれをMSX用に焼き直し，ことのついでにPC-6601の似非LOGO（BASICのプログラムで書かれたLOGO）のための解説本を書き散らし，勢いにまかせてSMC777のグラフィック入門書を書いた．

これが，私の出発のあらましだ．

正直に言えば，私は，パソピア７の解説書を書くことによって初めてパソコンというものを知ったのであるが，当時はそういう男にでも本を書くことができた．パソコン関連の本は，表紙さえついていれば，そこそこに売れるものであり，表紙を付け替えれば，もう一度新刊に戻ることができた．そういう状況の中で，私は，取りあえずの選択（そうでない選択があるだろうか？）として，この業界に身を置くことにした．ほかにどこに行くあてもなかったし，とにもかくにも，ねずみ車は回りはじめ，走り続けるほかに，私には選

択の余地がなかった．

私は，PSGによる6重の和音を出し，16色の色を同時に表示できたパソピア７を

「夢のマシン」

と書いた．

しかし，醒めない夢は，やはり存在しなかった．夢から醒めてみると，すべてのスペックはお笑い草だった．５インチ２Ｄのディスクはまたたく間に命を失い，パソピア７は静かに発売中止になった．

『親子で楽しむパソピア７』のおまけに，我々（私と，共同執筆者であるＨ野先生）は，カセットベースの販促用プログラムを書いた．それは，6つの和音を使いながら馬鹿なワルツを奏で，BASICのライン命令とペイント命令による間抜けなキャラクター（「パソ子」という名前の女だった）をのろのろと描く愚にもつかない代物であったが，ある日池袋西武の電気製品売り場に行ってみると，なんとその販促用プログラムが，エンドレスで実演されていた．

「おい，パソ子じゃないか」

私は驚愕した．

パソ子は，カタカナだけのセリフ（マア，コンナニ　タクサンノ　イロガ　デルノネ）を200ラインモニタに表示しながら，GET/PUT命令で目を白黒させていた．

このバカげた恥さらしのプログラムが白昼堂々人前で実行されていたのが，たった6年前のことだと思うと，私は頭がどうかしそうになる．

この6年の間，私は当てずっぽうのタイムスリップをしていたのであろうか．大学を出て，サラリーマンになって，退職し，失業者になって，本を書き，プログラムを書き，アブク銭を手に入れ，その金で呑んだくれて，次の本を書き，そうやって自分のまわりの状況が予期せぬ方向に転がって行くのに身をまかせているうちに，私はこのパソコン世界から決して逃れることのできない人間になってしまったのだろうか．

真面目な話，有為転変の激しいこの世界にいると，つくづく「死」というものについて考えさせられる．

死が不可避のものであるのだとしたら，我々の生は，そのまま緩慢な死にほかならないなどと，まだ何もしらなかった学生の頃，私はそういう抽象的なことを考えていたことがあるが，パソコン世界では，死はもっと具体的にして日常的なもので，犬のクソみたいにそこら中に転がっていた．しかも，ここで私が見てきた死は，緩慢な死どころか，すべて頓死であり，突然死であり，葬式も執り行われない，あっけらかんとした不条理な，間抜けな死ばかりであった．

「今日，ママンが死んだ」

そうだろうとも，ママンは毎日死んでいる．

新しいマシンが出る度に，我々は一応盛り上がる．

「おお，CD-ROMが標準装備か」

「へえ，ラップトップがついに20万円を切ったか」

そして，そういう新しいマシンの誕生に伴って，不可避的に死ななければならない（ここでは，子の誕生は親の死を前提としている）マシンについて思いを馳せる．

「おお，ついにVMにも寿命がやってきたか」

「ああ，あわれFM-オアシスよ」

しかし，本当のところ，私はいちいち「おお」とか「ああ」とか思っているわけではない．

美空ひばりが死んだ時にも，私は驚かなかった．手塚治虫が死んだ時にも，まるで意外に思わなかった．なにしろ

TK-80の昔

から，私はここに座っていて，無数の葬儀に立ち会ってきている．

「このたびは，どうもとんだことで……」

と，この業界に長いこと住んでいる人間の態度は，だんだん曖昧になる．最後にはベテランの葬儀屋みたいな鉄壁の無表情を身に付けるようになる．

ここまで読み返してみて，自分の文章の沈痛さに，我ながら少し驚いた．

常に前ばかりを見ているこの世界に長い間住んでいると，いつしか，後ろを振り返ることしかできなくなるのかもしれない．

まあ，いずれにしろ，私は自分の足元だけは見ない．だって自分が細いロープの上に立っているなんてことは，見るまでもなく，分かりきっているんだから．

Illustration
NOBUKI FUKUHARA

お詫び：著者急病のため「オダジマタカシのただいま悪戦苦闘中」は休載いたします．代わって「十年大昔，あるいは顧みられることのない腐敗する無機物の記憶」(『PC-PAGE4／僕らのパソコン10年史』収録，翔泳社）を掲載いたしました．

Macintoshで考えたこと
PersonalからIntimateへ

大座畑重光　マッキンテリジェンス(株)

1 「パーソナル」の未来

パーソナルコンピュータの「パーソナル」とは，いったいどのような意味あいをもっているのだろうか．子供や老人からソフトウェアエンジニアや最先端のコンピュータ科学者にいたるまで，非常に幅広い層に親しみをもって使われこと．あるいは，使うという意識や感覚のないまま，透明感をもって使うことができること．そして，使えば使うほど親しみがわいてはなせなくなるようなもの…．「パーソナル」の意味はさまざまに考えることができる．

さらに，これからの「パーソナル」は，両親と同じように誕生とともにつきあいがはじまり，各人の種々のデータを得て，それを解釈しユーザーを理解してくれるような，人間にフレンドリィなものという意味あいをもっていくだろう．それは，ユーザーの成長とともに成長していくようなものでなければならない．また，ペットのようにリアクティブでかわいらしくなければならないのかもしれない．

現在，数多くのファミコンやパソコンが家庭やオフィスにはいりこみ，多くの人々が利用できる環境が整いつつある．そのなかで，かなり若い年代からコンピュータとのつきあいが始まっている．自分でプログラミングなどしなくても，すでにたくさんのアプリケーションプログラムがあるので，「探す」だけで自分が欲しい機能が見つかることもあるかもしれない．この現象は，プログラムを自分で書ける人にとっては，作ったほうが早いか，探したほうが早いかというような問題をひきおこすことになるかもしれない．とにかく，コンピュータとのつきあいは始まっている．(＊3-5)

ここでは，未来に向けての新しいコンピューティングのスタイルがどのように変化していくのか，あるいはどのような方向性をもつべきなのかなどを一緒に考えてみたい．そして，現実との差を認識しながらも，プログラマとしてコンピュータの理想像に少しでも近づくためにはどうすればよいのかを考えてみよう．

2 リアクティブなものへのあこがれ

アラン・ケイ（米アップルコンピュータ社アップル・フェロー）というすばらしいコンピュータ科学者がいる．彼は主として『子供のためのコンピュータ』について長年研究してきた．彼の研究グループは，生後22か月のある女の子を対象として，6か月前から使っているMacintoshのユーザーインターフェースをどれほど理解しているかという調査を行った．調査は，その子の動作をビデオに撮って，それを分析する方法をとった．

ビデオの中でこの子は，MacPaintらしきアプリケーションを使い，自分の手のひらよりも大きなマウスをちょこちょこと動かして絵を書いたり，いろいろ操作していた．マウスのクリックやダブル・クリック，それにドラッギングができるだけでなく，ファイルを保存することや，オープンすることもできる．たぶん文字はまだ知らないと思うので，アイコンやそれがある場所が文字情報の代わりをしているのかもしれない．これだけの動作がごく自然にできるのには，まったく驚いた．

ビデオを上映しながらアラン・ケイは次のように説明している．「この子にとって，コンピュータはもうテクノロジーじゃない．生まれたときからそこにあるんだから腕時計や鉛筆やクレヨンと同じだ．非常に複雑なアプリケーションであるPageMakerのようなものも使えてマークをしたり，プリントアウトをしたり，さまざまな作業をすることができた．分析の結果，何とMacintoshユーザーインターフェースの70%を理解していることが判明した．コンピュータこそが真のメディアで，幼いときから一緒に成長していくことができるので，この世代には，コンピュータはもうテクノロジーではない．』（＊4）

生まれたときから，人間はリアクティブな（反応する）ものに興味を示す．これまで，生命体以外にはリアクティブなものとの付き合いはほとんどなかった．そのようなマシンがたとえあったとしても，それは非常に単純なものでしかなく，多様な反応を示すものとしてはコンピュータが初めてだ．

子供は，いちばん正直できびしいユーザーであるともいえる．赤ちゃんのときからコンピュータとの付き合いがはじまり，成長とともに自分自身のありとあらゆるデータを残せる可能性をもつ時代がやってきた．このことは未来において非常に重要である．なぜならば，個人が亡くなった後でもその人のデータ，情報が永遠に生き続ける可能性がでてくるからである．それがもとになって，その人の行動や推論が計算できるかもしれない．すなわち，その人の死後においても，その人のデータや情報に質問することが可能になるかもしれない．そして，その人の人生経験が無駄にならずに，後世に活かせる時代がくるかもしれない．

3 パーソナルを少しは実現できたMac

「パーソナルコンピュータ」という言葉は，1968年にARPAが主催した大学院生の会議で，当時ユタ州立大学の大学院生であったアラン・ケイがはじめて使ったといわれている．パーソナルコンピュータのことを「パソコン」と親しみをこめて呼ぶ人もいれば，なかば軽蔑して呼んでいる人々もいる．その名前は，「個人用のコンピュータ」を意味するが，「個人用」とはいったいどのような意味だろうか．

個人的には，「特定の個人とつきあえる相棒，コンピュータ」というふうに考えている．それは，主人であるユーザーのことをよく理解している．主人の性格や癖，しぐさ，知識，興味，友人関係，仕事の管理などの個人的な特性をよく知っていなければならない．さらに，その主人を取り巻く種々の環境のことも理解していなければならない．したがって，その特定のパーソナルコンピュータをユーザーである主人以外の人が使用するとなるとどうなるか．そのパーソナルコンピュータの応答がにぶくなったり，ちぐはぐなものになったり，要求に十分答えられないかもしれない．それでもいいと思う．このようなことが実現できるようになるには，パーソナルコンピュータのほうで，主人がだれであるかを理解できなくてはならない．そうなれば，ちょうど新米の秘書と経験を経た秘書とのサポートの差のようにふるまってくれる．パーソナルコンピュータのもつ知識は，どのような人に対しても共通な一般的な部分と，その主人に特有な部分から構成されていなければならない．

パーソナルコンピュータがこのような役割を果たすことができてはじめて，身近な道具，いやそれ以上のよき相棒的な関係が生まれるだろう．

Macintoshでさえ，まだこのようなものとは大きくかけ離れたもので，ソフトウェアの面ではいまだに原始時代であるといえる．

Macintoshにはデスクアクセサリというものがあって，そのなかにはコントロールパネルやスクラップブック，それにメモ用紙などのツールがある．コントロールパネルは種々の環境を個人にあうようにセッティングできる．スクラップブックやメモ用紙は，個人用の種々のデータを格納しておくことができ，いかなるアプリケーションのなかにいても，それらをアクセスすることができる．

4 身近なコンピュータへの試み

今後のコンピュータのひとつの発展方向は，メディアとして人間にとってより身近な存在になるものである．使うときにも存在感のない透明なものになる．コンピュータのパワーは，すべての人の行動を支援することのために使われるだろう．

MITメディアラボは，コンピュータを中心とする未来技術，メディアに関して，いま最も注目されている研究をしているところである．メンバーは多

種多様な分野におよび，百数十名で構成されている．ここでは芸術家と科学者がぶつかるのではなく，芸術と科学がぶつかり合い，反応を起こすことによって，すばらしいアイデアや発明が生まれることが期待されている．日本においても日本版のメディアラボ「メディア科学研究所」(東京都千代田区神田駿河台1-5-2，TEL 03-3259-0281（代））の設立が予定されている．もうすぐ，この研究所の母体となる「国際先端技術開発研究財団（IARDI」が設立されようとしている．MITメディアラボのネグロポンテ所長は，「日本の研究所では，メディアラボの研究そのものをもってくるのではなく，研究のスタイルを移植したい」と語っている．(＊2)

アップル社では，近未来に実現しようとしているものの具体的なイメージを公表し，これを「Knowledge Navigator」と呼び開発を進めている．このような理想的なスタイルを明確に打ち出している企業は，現在きわめて少ないのではないだろうか．スカリー会長はその実現性について，「技術的にはソフトウェアが問題で，とくに，コミュニケーション，ネットワーキング，データベースアクセス，ユーザーインターフェース，ソフトウェア開発環境，人工知能で，そのうち人工知能を除く部分については21世紀までに実用化できると思う」と語っている．

アラン・ケイいわく，「身近な（Intimate）コンピューティングという考え方は最終的には衣類，衣服，洋服のような形にならなければならない．たとえば，カリフォルニアにはTシャツの中にコンピュータを埋め込むという考え方がある．明日の形は，どこへでもつれて行ける，人間の行動に追従できるコンピュータで，常に世界的なネットワークに接続されているようなものである．そして，すべてのメディア，そして物理的な世界に存在しないようなメディアを全部統合し，すべてをダイナミックな形でリンクされていることが必要だ．」(＊4)

ビバリウム（Vivarium）という魅力的な名前のプロジェクトがロサンゼルスの公立の実験学校で進められている．ビバリウムとは，生命のための閉ざされた環境・生態系である実験用動物園のことを意味する言葉で，アタリ社にいたときにケイの同僚アン・マリオンが名づけた言葉である（＊1）．

ここでは300人の児童が勉強している．興味深いのは，動物の勉強のしかたについてである．たとえば，生態学，動物の行動，海の生態というような勉強をしている．子供たちに，ただビデオの資料などを与えるだけでなく，そのあとで魚や，それに関連する生物，植物が，生態系にどう寄与しているかを教えて，さらに実体験によって学ばせる方法をとっている．つまり，理論の確認を実際にやらせている．

また，モデル構築をさせ，このモデルが現実とどれだけ近いかを確認させている．このことにより，多様な異なる観点から勉強することができる．プロジェクトの中心は，プログラミング環境を子供たちのために作るということで，ほかの人がやったことをただ体験するだけでなく，子供たちがもっと積極的にシミュレーションをするようにしている．自主的に積極的にデザインをやるということが，もっとも効果的だと考えているからである．

4年前からこのプロジェクトを行っていて，子供たちのために新しい種類のユーザーインターフェースとAIプログラミング言語を構築し，ちょうど半分くらい進んだところだそうなので，1995年以後には完成するかもしれない．(＊4)

アップルフェローのアラン・ケイ博士やビル・アトキンソンは，Intimateコンピューティングに移行していかなければならないことを指摘している．そのアプローチとして，アラン・ケイはビバリウムプロジェクトを実践し，ビル・アトキンソンはPIC（Personal Intelligent Communicator）の開発に着手している．

5 かなり先のIntimateなものの追究

筆者は，情報を自由自在に航行でき，より柔軟なコミュニケーションを可能とするようなメディアのひとつとして，「航行めがね（NaviGlasses)」というものを提案している．(＊8)

「未来の装着型のコンピュータ，コミュニケーションのためのメディアの理想的なひとつの形，外見は“めがね”になる」というものだが，これは現在あるようなたんなるめがねではなく，現在のテクノロジーでは考えにくいような大規模なコンピュテーションパワーをもつことを前提にしている（図1，2参照）．

「航行めがね」のアイデアは，コンピュータ・スクリーン上に表示されたオブジェクトだけをクリックするという，従来の限られた世界から，それと同様なことを現実の世界まで拡張したいという願望から発想した．すなわち，現実のもの（オブジェクト）をクリックして，その情報をリアルタイムで得たいということだ．一種の疑似テレパシーのようなことを工学的に実現できるメディアである．航行めがねのレンズには，ピクセルに対応した画像入出力のための膨大な数のマイクロTVカメラがちりばめられ，それでいて全体としては透明である．ユーザーは，こ

のレンズを通して目に映る自然界の現実のもの（オブジェクト）に対してもアクセスし，コミュニケーションを行う．このことは，これまでのコンピュータによる情報のアクセスの対象を大幅に拡張し，人間を取り巻くコンピューティングを多様化させ，コンピュータにフレンドリィでない多くの人々にコンピュータのパワーを解放しようとするものである．

これは，人間へのサポートをより強力にする．これまでのコンピュータは，コンピュータディスプレイ上に表示されたデータやオブジェクトにしかアクセスできなかった．したがって，航行めがねはこれまでのコンピュータの計算形態では取り扱うことのできなかったことの一部を可能にするものである．これは我々が住む外界に対するコンピューティングの可能性を開く．このような，自然のなかに存在する生物や人造物に対するコンピューティングを，仮に「現実のオブジェクトに関するコンピューティングROC（Real Object Computing）」と名づけた．（*8-16）

このようなメディアが実現されれば，社会にたいしてきわめて大きな影響を与えると思われる．原子力にもまさるとも劣らないというほどに．原子力は大きなエネルギー源としては大歓迎されるが，武器としては歓迎されない．このように，よい面と悪い面をもっているが，航行めがねにおいても同じことがいえる．よいことだけのために使われることを望む．

6 おわりに

想像は果てしなく広がり，限りがないように思える．いろいろな角度からものを観察する習慣をつけ，実践していくことが大切であろう．

MacintoshのHackerのことを筆者はMackerと呼んでいる．Mackerは大きなリスクをかかえながらも，何か一貫した考え方をもち，独自のシステムを探究し，構築している．何か印象的な仕事を残して欲しい．

図1　航行めがねの外観

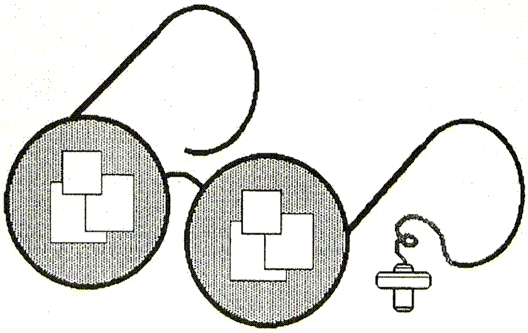
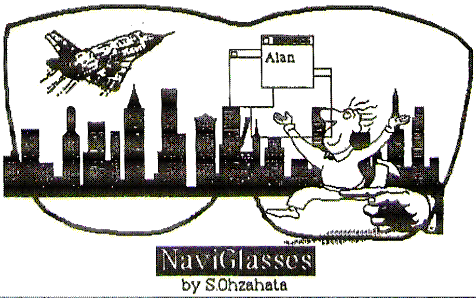


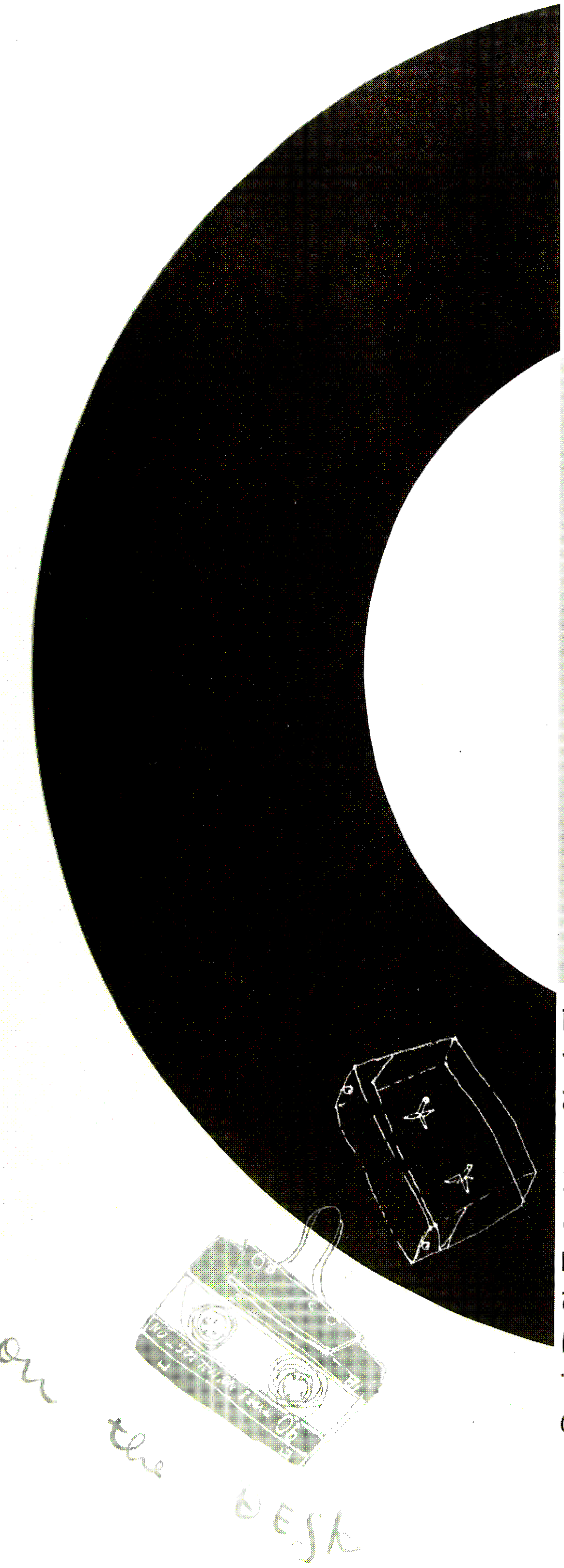
図2　航行めがねをかけたユーザーの視野



参考文献

*1　S.ブランド（室・麻生訳）：メディアラボ，1988，福武書店
*2　服部：使いやすいコンピュータをめざす，メディアラボー米MITに設立5周年ー，朝日新聞，1990年11月12日号
*3　大座畑：ヒューマン・コンピューティング環境の未来，MAC+CYBER，Vol.8，No.18，pp.42-49，佐久書房(1989.12)
*4　大座畑：アラン・ケイの講演，bit，Vol.22，No.2，pp.40-43，共立出版(1990.2)
*5　大座畑：ぼくらの時代のNEW COMPUTING STYLE-コンピュータが友だちになる日，THEソフトバンク'90，pp.16−19，日本ソフトバンク(1990.3)
*6　坪田：NCインタビュー，日経コンピュータ，pp.56-59，日経BP社(1990.10.22)
*7　大座畑：ビル・アトキンソンの旅，bit，Vol.23，No.1，pp.47-53，共立出版(1991.1)
*8　大座畑：これからのコミュニケーションメディアを考える(1)，情報を航行するメディア『航行めがね』の夢ー現実のもの(オブジェクト)をクリックしたいー，TURING MACHINE，Vol.2，No.3，pp.38-39，スペック（1989.6)
*9　大座畑：これからのコミュニケーションメディアを考える(2)，『航行めがね』プロジェクトー航行めがねとのインタフェース，TURING MACHINE，Vol.2，No.4，pp.56-58，スペック（1989.8)
*10　大座畑：航行めがねプロジェクト，人工知能学会ヒューマンインタフェースと認知モデル研究会，SIG-HICG-9001-5，pp.41-50（1990.6)
*11　大座畑：航行めがねの社会，『ソフトウェア科学・工学における数理的方法』(SSE)研究集会（9/27-28)(1990.9)
*12　大座畑：航行めがね：新しいコミュニケーションメディア，日本ソフトウェア科学会第7回大会論文集，C2-3，pp.77-80（1990.10)
*13　大座畑：航行めがねにおけるRAR概念：人工現実感と現実の世界の融合，第6回ヒューマンインタフェースシンポジウム（1990.10.24-26）論文集，pp.9-14，計測自動制御学会（1990.10)
*14　大座畑：航行めがね：RARのための究極のデジタル化をめざして，人工知能学会FAI，HICG，KBS合同研究会，SIG-F/H/K-9001-7，pp.57-66（1990.12)
*15　大座畑：航行めがね：RARの記述と背景となる情報環境，情報学シンポジウム，情報処理学会（1991.1）(予定)
*16　大座畑：航行めがね：人工テレパシーを実現するメディア，情報処理学会第30回全国大会講演論文集（1991）(予定)'

Illustration
MASAMI NOZOE



実践C塾 第2回

メモリモデルとオーバーレイ

塾頭 水居 浄

前回はMS-DOS上のCコンパイラをとりまく現在の状況について概略を説明しました．今回からは，代表的な処理系であるMicrosoft CとTurbo Cについて，それらの特徴を細かく検討することによって，MS-DOS上で実際にプログラミングする際のさまざまな問題点，およびそれらの解決方法などについて論じていきたいと考えています．

Microsoft CはV.5.1およびV.6.0を，Turbo CはV.2.0およびTurbo C++ V.1.0を対象とします．ただし，Turbo C++についてはTurbo Cの最新バージョンという視点で扱います．C++としての機能については本稿の目的から外れますので触れません．

ワンラインNEWS
内田洋行は学校向けパソコン教育システムの販売を12月に開始した．富士通のFM-TOWNSを使ったシステムで32,000色の豊富な色彩で画像の送受信ができるのが特徴．

1 Microsoft CとTurbo C

この２つの処理系はいろいろな意味で非常に似ています．というのも，後発のTurbo Cが登場する時点で，Microsoft Cがすでに業界標準としての地位を確立していたため，Turbo CがMicrosoft Cに対して，外部仕様上かなりの部分で互換をとったからです．しかし，細かい点では違う部分も多くあります．実際，オブジェクトコードレベルでの互換性は完全にはありません．両処理系の違いについて，まずコード生成という観点から検討したいと思います．

8086というCPUの性質上，Cコンパイラで作成するプログラムにはメモリモデルという概念があります．これは前回説明したように，表１のようなアドレスのサイズによる違いです．

以下，両処理系のメモリモデルの詳細について検討しますが，その前にいくつかの用語について明確にしておきます．

セグメント（segment）
複数の論理セグメントの集まりをグループと呼ぶ

前回は，１つのセグメントレジスタによって参照されるメモリ領域のことをセグメントと呼びましたが，正確にいうとこれは「物理セグメント」という概念です．8086のオブジェクトモジュールフォーマットには，これ以外に「論理セグメント」という概念があります．

論理セグメントとは，１つの機械語データ（プログラムコードでもデータでもなんでもよい）の集りを表す最小単位です．すべてのプログラムは１つ以上の論理セグメントの集りとして表されます．この論理セグメントは，名前や属性をもちます．リンカはこの名前と属性を参照して，各論理セグメントをプログラム中の所定の位置に配置します．

一方，物理セグメントは実際にメモリ上で１つのセグメントレジスタを使って参照されるメモリ領域です．複数の論理セグメントを集めて，１つの物理セグメントに配置することが可能です．このような場合，その複数の論理セグメントの集りを「グループ」と呼びます．つまり，１つのグループは１つの物理セグメントに対応するわけです．グループも名前をもちます．もちろん，１つの論理セグメントがそのまま１つの物理セグメントに対応することもあります．この場合，論理セグメントはグループ化されません．

たとえば，Microsoft Cのスモールモデルで作成したプログラムは，表２のような論理セグメントとグループをもちます．

ここでは，コード部は１つの論理セグメントがそのまま１つの物理セグメントに対応しています．一方，データ部は複数の論理セグメントが集まってDGROUPという名前のグループを構成し，これが１つの物理セグメントに対応しています．

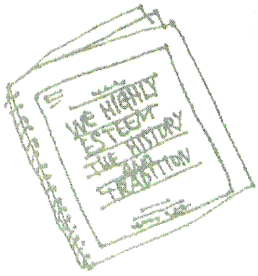
表１　メモリモデル

メモリモデル	コードアドレス	データアドレス
スモール，タイニィ	near	near
ミディアム	far	near
コンパクト	near	far
ラージ	far	far
ヒュージ	far	huge

表２　Microsoft C（スモールモデル）の論理セグメントとグループ

	論理セグメント名	グループ名
コード部	_TEXT	(なし)
データ部		
静的データ（初期化あり）	_DATA	DGROUP
静的データ（初期化なし）	_BSS または c_common	DGROUP
静的データ（const型）	CONST	DGROUP
スタック	STACK	DGROUP

注：実際にはこれら以外にもいくつかの論理セグメントが定義されていますが，それらはランタイムルーチンが内部的に使用するもので，ユーザーの定義したデータやコードとは直接関係ありません．



静的データ (static data)
静的データとは，変数と文字列定数

静的データとは，externやstaticの記憶クラスで定義された変数と，文字列定数です．これは，次の４つに分類されます．

(1) 初期化された静的変数

定義と同時に初期化されたextern変数またはstatic変数です（図１）．

図１　初期化された静的変数の例

```
static int a = 123;
```

(2) 初期化されていない静的変数

定義時に初期化されていないextern変数またはstatic変数です（図２）．

図２　初期化されていない静的変数の例

```
static int a;
```

初期化されていないextern変数を，コモン変数と呼びます．Microsoft Cでは，コモン変数とそれ以外の初期化されていないstatic変数を別々の論理セグメントに配置します．

(3) const型変数

const型修飾子を使って定義されたextern変数またはstatic変数です（図３）．

図３　const型変数の例

```
static const int a = 123;
```

(4) 文字列定数

文字列定数です（図４）．

図４　文字列定数の例

```
char *a = "ABCDEFG";
```

図４の右辺は文字列定数です．この構文ではポインタａをそのアドレスで初期化しています．つまり，ポインタａにはメモリ上に置かれた文字列定数のアドレスが代入されるわけです．図５のような配列の初期化との違いに注意してください．

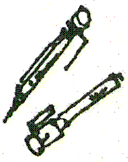
図５の場合，配列ａが右辺の文字列の内容で初期化されるだけで，文字列定数は生成されません．

図５　配列の初期化

```
char a[] = "ABCDEFG";
```

なお，一般の文字型や数値型の定数は，プログラム内で使われた場合，8086のイミディエートアドレッシングによって，直接プログラムコード中に埋め込まれます．ただし，Microsoft Cでは浮動小数点型の定数は，静的な定数データとして文字列定数と同様に扱われるようです．

静的変数に関しては，初期化されているかどうかが重要です．初期化された変数は，その初期値を実行形式ファイル中にそのままの形でもつ必要がありますが，初期化されていない変数は，起動時に０でクリアすればよいので，実行形式ファイルの大きさを節約するようにすることが可能だからです．この処理のため，一般に初期化された静的変数と初期化されていない静的変数は別々の論理セグメントにわり当てられます．



スタック (stack)
SSレジスタを使ってわり当てる物理セグメント

スタックは，8086のPUSH～POP命令でレジスタなどの値をセーブ／リストアしたり，CALL命令で呼び出されたサブルーチンから復帰するアドレスを保存したりする目的で使用されるメモリ領域です．Cでは一般に関数呼出しはCALL命令になりますが，その際リターンアドレスだけでなく，引数もスタック上に置きます．呼び出された関数は，引数の値をスタック上から取り出して使用するわけです．また，自動変数も一般にスタック上に置かれます．自動変数はその関数の内部だけで使用するものですから，関数に入ったときにスタック上に領域を確保し，関数から抜けるときにスタック上から取り除くようにすれば，メモリを有効に活用できるのです．もちろん，引数や自動変数をスタック上に置くことには，再帰呼出しや再入を可能にするという重要な目的もあります．

8086では，スタックにはSSレジスタを使って専用の物理セグメントをわり当てることができます．無論ひとつのセグメントですから，最大64KBまでという制限が付きます．さらに，スモールモデルやミディアムモデルでは，別な制限が出てきます．それは，DSと

ワンラインNEWS
明電舎は統合化，リアルタイム性をもたせたFAコンピュータを開発した．同機は日本電気の32ビットCPUを搭載，OSにはUNIXとITRONを組み合せた統合型OSを採用．

SSの値を同じにしなくてはならないということです．スモール／ミディアムモデルでは(farキーワードなどを付けないかぎり)，データはすべてnearアドレスをもちます．前述のように，自動変数や引数はスタック上，すなわちSSレジスタの指すセグメントに置かれます．たとえば，図6のような静的変数aと自動変数bおよびポインタ変数pを考えます．

図6　静的変数a，自動変数b，ポインタ変数p

```
static int a;
auto int   b;
int        *p;
```

この場合，ポインタpにはaのアドレス&a，bのアドレス&bのどちらでも正しく代入できなくてはなりません．無論，これらのアドレスはすべてnearアドレスです．つまり，静的変数は自動変数と同じ物理セグメント，すなわちスタックと同じ物理セグメントに置かなくてはならないのです．

また，8086ではデータへアクセスする命令は，大多数がDSレジスタの指すセグメント上のデータを前提にしています．SSレジスタを使ってアクセスすることも可能ですが，コードが大きく遅くなります．したがって，静的変数などの静的データはすべてDSレジスタの指すセグメント上に置かれます．この結果，一般にスモール／ミディアムモデルではDS==SSにする必要性があるのです．

もちろん，同じ1つの物理セグメントに静的データとスタックが同居することになりますから，スタックとしてどれだけのメモリ領域を使用することができるかは，静的データと後述するヒープ領域をどれだけ使用するかに依存してしまいます．したがって，スモール／ミディアムモデルでは，あまり大きなスタックを確保することはできません．デフォルトのスタックサイズは，Microsoft Cでは2KB，Turbo Cでは4KBとなっています．無論，この大きさは変更できます．

以上のような違いがありますから，ヒープと静的データ，自動変数はそれぞれ目的に応じて使い分けなくてはなりません．なお，スモール／ミディアムモデルでは，スタックと同様にヒープもDS==SSの指すセグメントに置かれなくてはなりません．ヒープ上のデータもnearアドレスをもたなくてはならないからです．すなわち，スモール／ミディアムモデルでは，ひとつの物理セグメントを静的データ，スタック，ヒープの3つで使い分けることになります．

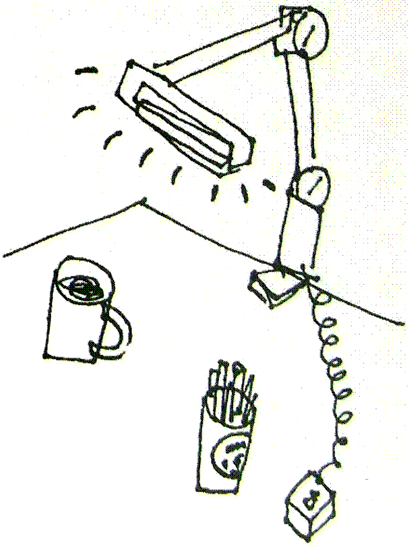
次に，各メモリモデルについてMicrosoft CとTurbo Cを比較しながら検討します．

ヒープ（heap）
動的にわり当てられるメモリ領域

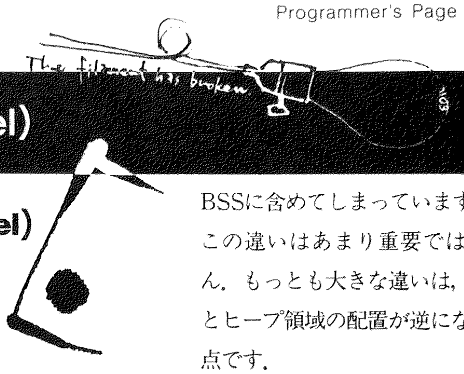
ヒープとは，動的にわり当てられるメモリ領域です．静的データはプログラムがメモリ上にロードされた時点から存在している領域ですが，ヒープ上のデータは，プログラムの実行中に領域をわり当てられます．ヒープ上に領域をわり当てるには，一般に標準ライブラリ中のmalloc関数やcalloc関数などを使います．mallocやcallocでわり当てられた領域は，free関数を呼び出すと解放（放棄）されます．

静的データではなくヒープを使う利点は，領域の大きさをプログラムの実行時に指定することができ，しかも解放した領域はまた違う目的に再利用できるという点です．逆に欠点としては，領域を確保するためにいちいちプログラム中でmallocを呼び出さなくてはならない点や，要求した大きさの領域をつねに確保できるわけではないので，エラーチェックそのほかのコードを書かなくてはならないなど，プログラムが多少複雑になる点です．

一方，動的にわり当てられるという点では，スタック上に置かれる自動変数も同じです．しかし自動変数の場合は，宣言した関数から抜けるときに自動的に領域が解放されてしまいますが，mallocでわり当てた領域はfree関数を呼び出さないかぎり解放されません．



2 メモリモデル（memory model）

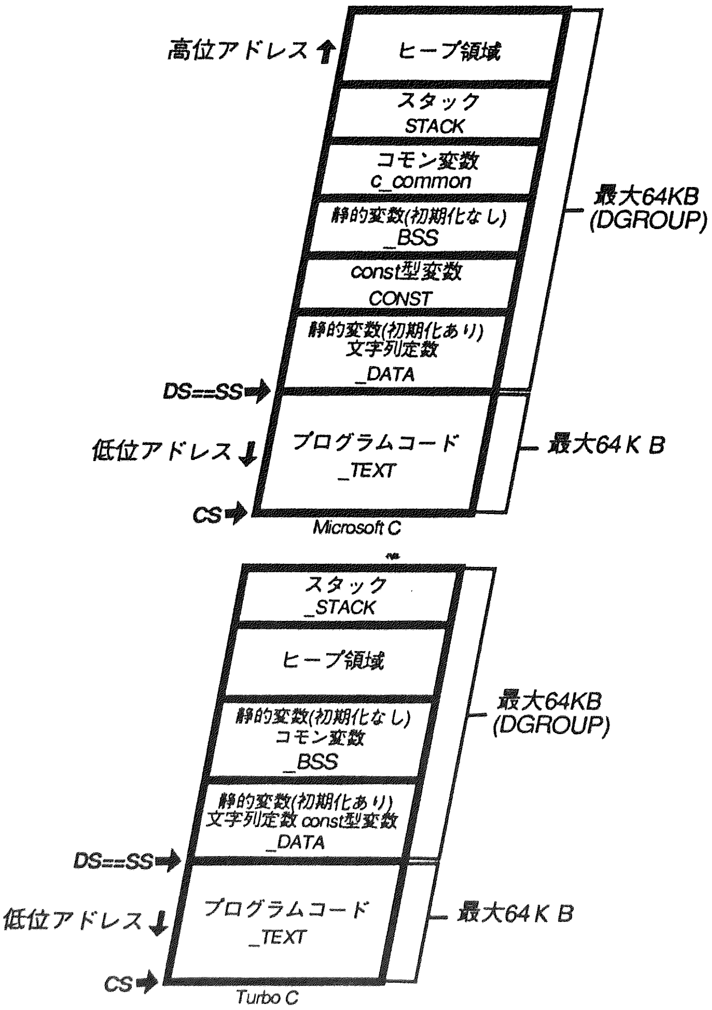


スモールモデル（Small memory model）コード，データにそれぞれ1つのセグメントを配置

もっとも基本的なメモリモデルです．1つのコードセグメントと1つのデータセグメントから構成されます．コードセグメントにはプログラムコードだけが置かれますが，データセグメントには静的データとスタック，ヒープ領域が置かれます．

図7に示すように，Microsoft CよりTurbo Cのほうがシンプルな構成になっています．すなわち，const型変数やコモン変数に固有の論理セグメントをわり当てず，それぞれ_DATAと_BSSに含めてしまっています．しかし，この違いはあまり重要ではありません．もっとも大きな違いは，スタックとヒープ領域の配置が逆になっている点です．

図7　スモールモデルの構造



Microsoft Cでは，スタックより高位のアドレスにヒープ領域を置いています．したがって，64KBのデータセグメントから，静的データとスタックを除いた残りの領域をすべてヒープ領域として活用することができます．実際，Microsoft Cのスモールモデルで作成したプログラムは，起動時にまず64KBフルのデータセグメントを確保してしまい，mallocはそのあらかじめ確保したデータセグメント上に領域を取得するようになっています．したがって，Microsoft Cのスモールモデルで作成したプログラムはつねに64KB以上の大きさをメモリ空間上で占有することになります．たとえプログラム中でmallocを使っていなくてもそうなります（無論，空きメモリが不足しているときはこのかぎりではなく，データセグメントには取れるだけの大きさの領域を取ります）．これは通常は問題になりませんが，たとえばプログラムから子プロセスとしてほかのプログラムを起動するときには領域のむだが問題になります．

しかし，このむだについては回避手段があります．それには，Microsoft社のリンカLINK. EXEの/CPARMAXALLOCオプションを使います．リンク時に，図8（1）のように指定します．CL. EXEでコンパイルからリンクまで一括して行うときは，図8（2）のように指定します．このオプションは，実行形式ファイルが獲得するメモ

ワンラインNEWS
韓国の三星電子は消費電力の小さいDRAMである1M疑似スタティックRAMを開発，サンプル生産に入った．疑似SRAMはノートパソコンを中心に需要が拡大している．

図8　リンク時の指定

```
(1)  link /NO1 /CP:1 prog.obj ;
(2)  cl prog.c /link /CP:1
```

リ領域の最大値(maxalloc)を設定するものです．ここで最小限必要な大きさ(minalloc)より小さい値を指定すると，最小限必要な大きさと同じ値を自動的に設定してくれますから，オプションの引数には1を指定するわけです．

なお，余談ですが，この/CP:1を指定する場合は，リンカに同時に/EXEPACKオプションを指定したり，できあがった実行形式ファイルにEXEPACKユーティリティを使用したりして圧縮するのは避けたほうが無難です．これを行うと，実行形式ファイルのヘッダの情報が一部狂って(maxallocがminallocより小さい値になってしまうことがある)しまい，正常に実行できなくなることがあるからです．リンカおよびEXEPACKのバグと思われ，Microsoft C V.3.0〜V.5.1で発生します．Microsoft C V.6.0では修正されているようですが，確証はありません．

Turbo Cでは，スタックをヒープ領域より高位のアドレスに置いています．したがって，あらかじめどれだけの領域をヒープ領域として確保するかを，プログラム作成時に指定しておかなくてはなりません．デフォルトでは，ヒープ領域の大きさは可能なかぎり確保するという設定になっています．つまり，Microsoft Cと同様にプログラムの起動時にまず64KBのデータセグメントを確保してしまってから，スタックをその最高位アドレスに配置するのです．mallocは静的データとスタックの間に領域を取得します．よって，Microsoft Cと同様に，スモールモデルで作成したプログラムはつねに64KB以上の大きさをメモリ上で占有するという問題が生じます．

Turbo Cではこの問題を解決するためには，外部変数_heaplenを使用します．プログラムの先頭部分で，図9のように記述して指定します(初期化の構文で指定します)．

図9の例では，ヒープ領域の大きさを16KBとしています．もちろん，指定した大きさの領域が必ず確保できるとはかぎりません．確保できなかった場合は，プログラムは異常終了します．

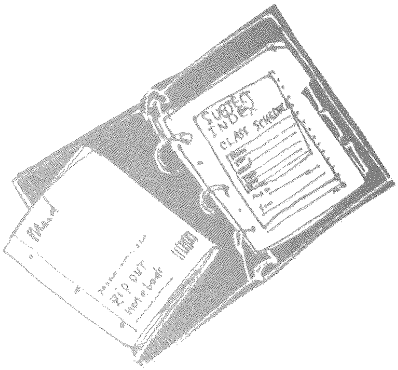
8086のスタックは低位アドレスへ向って伸びるようになっています．つまり，PUSHするたびにスタックポインタSPの値がデクリメントされます．このため，Turbo Cのスタックとヒープの配置だと，スタックの大きさを疑似的に可変にすることができます．ヒープをあまり使っていないときには，その領域をスタックとして利用することができますし，その逆も可能です．つまり，ひとつの領域を柔軟にスタックとヒープで使い分けることが可能なのです．Microsoft Cの配置ではこういうことはできません．実はMicrosoft CでもmVARSTCK.OBJ(mはメモリモデルS，M，C，Lのどれか)というオブジェクトファイルをリンクすることによって，スタックの使われていない部分をヒープ領域として使うことはなんとか可能です．しかしこの場合もスタックの配置を変更するわけではありませんから，ほかのヒープ領域と連続したヒープ領域としては使えませんし，ヒープ領域の使われていない部分をスタックとして使うこともできません．

図9　外部変数_heaplenの指定

```
extern unsigned _heaplen = 16384;
```

では，Turbo Cの配置のほうが絶対的に優れているのかというと，実はそうともいえないのです．Microsoft Cでは，前述の/CP:1オプションを使用するとプログラムの起動時に確保するヒープ領域をまったくなくすことができます．しかし，この場合でもヒープ領域を使用することは可能です．この場合，malloc関数はDOSに必要なメモリを要求して，ヒープ領域を新規に獲得するからです(無論，スタックのすぐ上の物理メモリがほかのプログラムなどに使われていない場合にかぎります)．このため，Microsoft Cでは，子プロセスを起動するときに不要なヒープ領域をfree関数で解放してしまうことで，メモリを有効に活用できます(実は，V.5.1までのMicrosoft CではfreeしたメモリをDOSに返却することができないという問題があるため，必ずしもメモリの有効活用にはつながらないのですが，この件についてはライブラリ関数について論じるときに検討します)．

いずれにせよ，このスタックの配置の違いはどちらの方法が優れているというわけではなく，設計思想の違いであるといえます．



ラージモデル（Large memory model）
コードもデータも複数のセグメントに配置

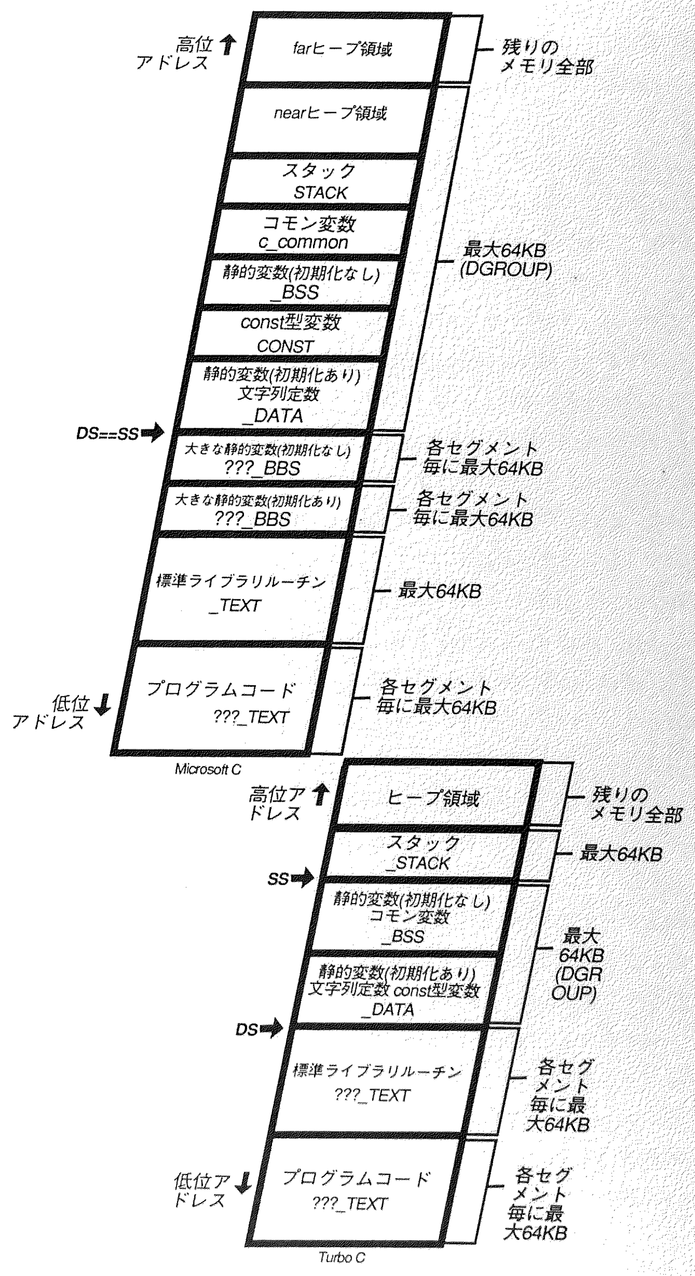
ラージモデルは，コードもデータも複数のセグメントを使用できるメモリモデルです．すなわち，関数呼出しはfarコールになりますし，データのアドレスもfarアドレスになります．

図10に示すように，プログラムコードには各ソースファイルごとに論理セグメントがわり当てられます．セグメント名はソースファイル名から.c拡張子を除いたものに_TEXTをつけたものがデフォルトで使用されます．もちろん，このセグメント名はユーザーがコンパイラのオプションスイッチで指定することも可能です．基本的に，これらの論理セグメントはすべてそのまま物理セグメントに対応します．すなわち，グループ化されることはありません(ただし，MS-LINKの/PACKCODEオプションを使うと，隣接するコードセグメントを合計64KBに達するまでまとめることができます)．

ラージモデルのプログラムコードにおけるMicrosoft CとTurbo Cの最大の違いは，標準ライブラリのセグメント名にあります．Microsoft Cでは，すべての標準ライブラリルーチンは，_TEXTというセグメント名で作成されています．一方，Turbo Cではユーザープログラムと同様に各ソースファイルごとに個別のセグメント名が与えられています．

この違いは実用上はそれほど重要ではないのですが，Microsoft Cの方式だと，場合によっては致命的な問題が起こる可能性があります．つまり，標準ライブラリルーチンがすべて_TEXTという同じ名前のセグメントにわり当てられているため，ラージモ

図10　ラージモデルの構造



ワンラインNEWS
任天堂のスーパーファミコンが昨年末に発売され，出足も快調で品不足が続いている．しかし従来のファミコンからのユーザーが移行しているだけともいわれている．

デルであるにもかかわらず，合計で64KBを超える量の標準ライブラリルーチンをひとつのプログラムで使用することができないという問題です．実際，標準ライブラリには300個を超える関数があり，それらの大きさの合計はかるく64KBを超えています．

Turbo Cでは，こういう問題は本質的に発生しません．では，なぜMicrosoft CはTurbo Cと同様の方式にしなかったのでしょうか？ それは，セグメントを同じにすることにより性能の向上が期待できるためだと思われます．標準ライブラリ関数は，たいていの場合，内部的にさらにほかの関数を呼び出して処理を行います．とくに，ライブラリの外からは見えない内部的な下請け関数を呼び出している場合が多くあります．そういう場合，セグメントが同じであれば，下請け関数の呼出しをnearコールにして性能の向上を図ることができるのです．

もっとも，いくら性能のためとはいっても，前述の問題はやはり残ります．しかし，実際にこの問題に突き当って困ったという話は，聞いたことがありません．そもそも64KBを超えるような，大量の標準ライブラリルーチンをリンクするようなプログラムを記述するケースは，現実にはほとんど発生しないということなのでしょう．

図10に示すように，データについても合計が64KB未満という制限がなくなり，複数のデータセグメントが配置されます．しかし，Microsoft CでもTurbo Cでも，特別なデータセグメントとしてのDGROUPは存在します．

Microsoft Cでは，DGROUPの構成はスモールモデルの場合と基本的に変わりません．スタックはあいかわらずDGROUPに含まれますし，静的データも大きさの小さいものはDGROUP内に配置されます．比較的大きな静的データは，新たにセグメントをわり当てて配置されます．この大きさの境界は，/Gtオプションで指定できます．デフォルトは32KBで，この大きさより大きな配列などが，DGROUPとは別のセグメントに配置されます．ただし，明示的にfar宣言したデータの場合は，この境界値に関係なくDGROUP外に配置されるようです．セグメント名は，ソースファイル名に任意の数字と_DATAをつけたものが使用されます．

一方，Turbo Cでは，ラージモデルでも静的データはすべてDGROUPに配置されます．このため，静的データの合計量が64KBを超えることができないという制限が生じます．ヒープに関して64KBの制限を受けないとはいえ，静的データが制限を受けるのでは，完全なラージモデルであるというのに抵抗を感じるところです．もっとも，最新版であるTurbo C++ではこの制限が解除されており，明示的にfar宣言した静的データは，個別のセグメントに配置されるようになっています．ただし，Microsoft Cと違って，データの大きさによって自動的に配置してくれるわけではありません．DGROUP外に追い出すものにはfarをつけて，プログラマが指定する必要があるわけです．この際，セグメント名はソースファイル名に任意の数字と_FARを付けたものが使用されます．

スタックは，Turbo CではDGROUPとは別になります．この結果，ほかの静的データの量にかかわらず最大64KBまでのスタックを確保することが可能です．Microsoft Cでは，スモールモデルと同様にDS==SSの制限があり，スタックはDGROUP内に配置されます．データアドレスがfarなのですから，Turbo C同様に独立したスタックを配置してもよいはずなのですが，Microsoft Cではそれよりもスタック上をnearアドレスでアクセスすることが可能なことによる，性能向上の可能性を重視したようです．

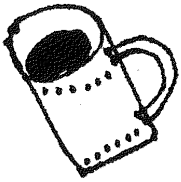
ヒープに関しては，Turbo Cは単純にスタックより高位アドレスの領域ということにしています．しかし，Microsoft Cではnearヒープとfarヒープのふたつがあります．nearヒープは，スモールモデルと同様にDGROUPの残りの部分をヒープ領域とするもので，farヒープはDGROUPより高位アドレスの領域です．このように分けているのは，nearヒープに確保したメモリはnearポインタを使って操作できるため，プログラムの作り方によっては性能向上に役だつからでしょう．ラージモデルのmalloc関数はfarヒープからメモリを獲得するので，nearヒープからメモリを獲得するには，_nmallocという関数を使用します．Microsoft Cは，ラージモデルでもやはり起動時にDGROUPに64KBフルのセグメントを確保してしまいます．スモールモデルの場合と同様にリンカに/CP:1オプションを与えることでこれは回避できますが，その場合，nearヒープ領域はなくなり，_nmallocはつねにエラーを返すようになります．

タイニィモデル（Tiny memory model）
コード，データを同一のセグメントに配置

タイニィモデルは，コードとデータを同一のセグメントに配置したスモー

ルモデルです．全体を単一のセグメントにすることによって，.COM形式の実行形式ファイルを作成することができます．

Microsoft CもTurbo Cも，タイニィモデルでは_TEXTセグメントをDGROUPに含めることで単一セグメント化したという点以外は，スモールモデルとなんら違う点はありません．なお，Microsoft CではV.5.1までのバージョンではタイニィモデルをサポートしていませんでした．V.6.0からはサポートしています．



ミディアムモデル（Medium memory model）
コードを複数，データをひとつのセグメントに配置

ミディアムモデルは，プログラムコードについてはラージモデルと同じです．したがって，ラージモデルと同様の特徴をもっています．一方，データについてはスモールモデルと同じになります．やはりスモールモデルと同じ特徴をもちます．

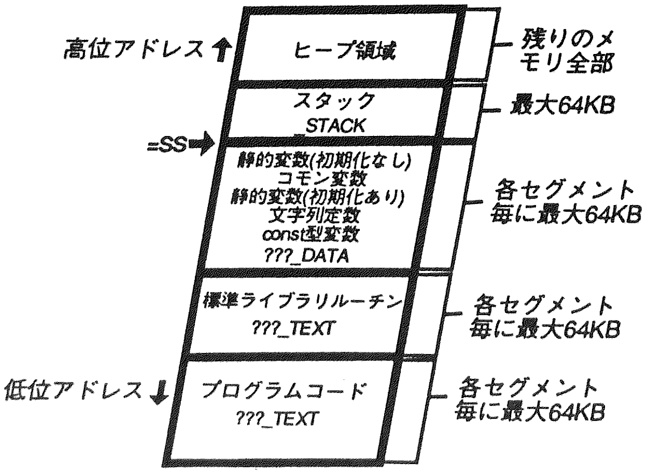
コンパクトモデル（Compact memory model）
コードをひとつ，データを複数のセグメントに配置

コンパクトモデルは，ミディアムモデルの反対の構成のメモリモデルです．すなわち，プログラムコードについてはスモールモデルと同じですが，データについてはラージモデルと同じです．

ヒュージモデル（Huge memory model）
巨大データの利用が可能

ヒュージモデルは，基本的にラージモデルと同じくコードもデータも複数のセグメントを使用できるメモリモデルですが，さらに単一のデータ項目が64KBを超える大きさをもつことを許すものです．

図11 Turbo Cのヒュージモデルの構造



Microsoft Cのヒュージモデルのセグメント配置は，ラージモデルとまったく同じです．64KBを超えるようなデータ項目（hugeデータ項目）は，静的データの場合は連続した複数のセグメントに分割して配置されます．ヒープ上にhugeデータ項目を獲得する場合は，halloc関数を使用します．

Turbo Cのヒュージモデルでは，図11に示すように，静的データはそれぞれソースファイル名に_DATAをつけた名前の論理セグメントに配置されます．これらはすべて別々の物理セグメントになり，DGROUPは存在しなくなります．しかし，hugeデータ項目は静的データ項目として確保することができず，ヒープ上にしか確保できません．ヒープ上にhugeデータ項目を獲得するには，farmalloc関数を使用します．Turbo C++では，静的データとしてhugeデータ項目を確保することができますが，hugeキーワードを明示的に指定する必要があります．

ワンラインNEWS
日本電気が昨年末に発売したPC-9801NVに搭載されているCPUはV30HL．このCPUはV30より消費電力を押さえ，高速化を実現したチップである．

混合メモリモデル（Mixed memory model）
アドレス修飾によるメモリモデル

以上，標準的な6つのメモリモデルの構造について説明しました．しかし，Microsoft CもTurbo Cも，これらのメモリモデルのデフォルトと違うアドレスサイズのデータやコードを使用することが可能です．前回説明したアドレス修飾の機能により，たとえばスモールモデルでfarデータを操作したり，ラージモデルで関数をnearコールしたりすることが可能です．リスト1のプログラムをMicrosoft CまたはTurbo C++のスモールモデルでコンパイルすると，配列fstrはDGROUP以外に配置されます（Turbo C V.2.0では静的なfar配列の宣言が許されていませんので，このプログラムはエラーになります）．以下，この混合メモリモデルのテクニックについて説明します．

スモールモデルをベースにした混合メモリモデル

現実に作成するプログラムは，よほど大がかりなものでないかぎり，たいていスモールモデルで作ることができます．とくに，混合メモリモデルの機能が登場して以来，かなり大きなプログラムでもスモールモデルで記述できるようになりました．

スモールモデルをベースにした混合メモリモデルでは，おもに次のようなテクニックを使います．

（1）大きなデータや高速アクセスを要求されないデータをfar定義して，DGROUPの外に追い出します．
（2）DGROUPに入り切らないような大きなバッファ領域などは，Microsoft Cでは_fmallocやhalloc，Turbo Cではfarmallocを使って，DGROUP外部に動的に確保するようにします．
（3）高速な実行を要求されない関数をfar定義して，_TEXTの外に追い出します．この場合far定義する関数は，ほかの関数とは別のソースファイルにして，さらにコードセグメント名を変更するオプションスイッチを使ってコンパイルする必要があります．

スモールモデルをベースにした混合メモリモデルを使うメリットは，速度的に重要でないコードやデータを，デフォルトのセグメントから追い出すことによって，全体の速度を普通のスモールモデルとあまり変わらないものにすることが可能な点です．しかし，逆に次のような制限もあります．

（1）DGROUPから追い出したデータに関しては，標準ライブラリ関数で直接処理することがほとんどできません．これは，スモールモデルの標準ライブラリ関数が受け取るデータアドレスが，nearアドレスに限定されるためです．たとえば，far定義した文字列をstrcpyなどで処理するようなことはできません．実際，farアドレスをサポートしているのは，グラフィックライブラリなどの特別なものを除けば，printf系関数の"%F"指定ぐらいしかありません（ただし，Microsoft C V.6.0ではfarデータ用の文字列操作関数などが追加されています）．
（2）_TEXTから追い出したコードから，標準ライブラリ関数を呼び出すことができません．これは，スモールモデルの標準ライブラリ関数はnearコールで呼ばれることを前提にしているため，_TEXTの外から呼び出すとかえってこれなくなるからです．ただし，グラフィックライブラリの関数は元々farコールで呼ばれるように作られているため呼ぶことは可能です．
（3）（2）と関連したことですが，_TEXTから追い出した関数では，スタックチェックを禁止しなくてはなりません．スタックチェックルーチンはほかの標準ライブラリ関数と同様にnearコールされることを前提にしているためです．同様に，long型の乗除算や，浮動小数点型の演算も行ってはなりません．これらの演算は内部的な演算用のサブルーチン（コードヘルパ）の呼出しとしてコード生成され，やはりそれらのルーチンがnearコールを前提にしているためです．

これらの制限は，かなり厳しいものに見えます．実際，標準ライブラリ関数をプログラム全体にわたって使っているようなプログラムでは，これらを採用するのは容易ではないと思われます．しかし，標準ライブラリ関数をあ

リスト1　スモールモデルでのfarデータの使用例

```
char far fstr[] = "Hello, world.";

main()
{
    printf("%Fs¥n", fstr);
}
```

まり多くの場所で使わないプログラムでは，使う部分と使わない部分とを切り分けていくことで，うまく対応することが可能です．すなわち，スモールモデルをベースにした混合メモリモデルは，どちらかというと上級者向きの高度なテクニックであるといえます．

リスト2に，混合メモリモデルプログラムの非常に簡単な例を示します．これはPC-9801のテキストVRAMに直接文字列を書き込むものです．このプログラム自体は，大きさからいっても混合メモリモデルにする必要性はまったくないのですが，例として示すためにあえてそうしてあります．

リスト2　スモールモデルをベースにした混合メモリモデルプログラムの例

```
/* main.c */
void far cls(void);
void far print(int x, int y, char *s);

main()
{
    cls();
    print(30, 10, "Hello, world.");
}


/* sub.c */
unsigned short far *tvram = (unsigned short far *)0xA0000000L;
unsigned short far *avram = (unsigned short far *)0xA2000000L;
int pos;

void far cls(void)
{
    int i;
    for (i = 0; i < 80*24; ++i) {
        tvram[i] = ' ';
        avram[i] = 0xE1;
    }
    pos = 0;
}

static void printchr(char c)
{
    tvram[pos] = c & 0xFF;
    avram[pos++] = 0xE1;
}

void far print(int x, int y, char *s)
{
    pos = y*80 + x;
    while (*s != '¥0')
        printchr(*s++);
}
```

Microsoft Cの場合，次のようにしてコンパイル・リンクします．

```
cl /c main.c
cl /c /Gs /NTSUB_TEXT sub.c
cl main.obj sub.obj
```

Turbo Cの場合，次のようにしてコンパイル・リンクします．

```
tcc -c main.c
tcc -c -zCSUB_TEXT sub.c
tcc main.obj sub.obj
```

ラージモデルをベースにした混合メモリモデル

大規模なプログラムを作成するには，ラージモデルを採用するのがもっとも簡単な方法です．しかし，ラージモデルでは，すべてのデータがfarアドレスを使ってアクセスされますし，すべての関数がfarコールされてしまいます．ここでよく考えてみると，たとえばDGROUPに配置されていることがわかっているデータは，nearアドレスでアクセスしても不都合はありません（コンパクトモデルやラージモデルにおけるnearアドレスは，DGROUP上のオフセットアドレスを意味します）し，また呼出し側と同じソースファイル中に記述された関数は，ほかのソースファイル中の関数から呼び出されないかぎり，nearコールしても不都合はありません．

ラージモデルをベースにした混合メモリモデルでは，このようなソースレベルでの最適化を行うことが目的になります．この場合，スモールモデルをベースにした場合のような制限はありません．ラージモデルの標準ライブラリ関数は，すべてのデータアドレスをfarアドレスとして受け取りますし，またすべてfarコールされるからです．nearアドレスをfarアドレスに変換しても，必要な情報を失うことはありませんから，きちんとキャストさえしてやれば問題なく処理できるわけです．したがって，ラージモデルをベースにした混合メモリモデルは，比較的容易に使用することができます．

ほかのメモリモデルをベースにした混合メモリモデル

ミディアムモデルをベースにした場合は，コードについてはラージモデルの場合と同様で，データについてはスモールモデルと同じメリットと制限があります．コンパクトモデルをベースにした場合は，逆にコードについてはスモールモデルの場合と同じメリットと制限があり，データについてはラージモデルと同様です．

ワンラインNEWS
日立家電販売は英和・和英翻訳機能を標準装備した電子手帳を12月に発売した．同電子手帳の本体はシャープからOEM供給を受ける．

モデル依存しないライブラリ (Model-independent library)
全メモリモデルで使用可能なユーザーライブラリ

混合メモリモデルのテクニックを応用すると，メモリモデルに依存しない，すなわちすべてのメモリモデルで共用可能なユーザーライブラリを作成することも可能です．これは，次のような方法で作成します．

(1) ライブラリの各関数は標準ライブラリの関数をいっさい呼び出さないようにします．これはスタックチェックやlong型の演算などの隠れた呼び出しも禁止することを意味します．
(2) 各関数はfarコールで呼び出されるようにします．データのアドレスもfarアドレスで渡されるようにします．
(3) プログラムコードには_TEXT以外の，独自のセグメント名を与え，またライブラリ内部で使用するローカルな静的データもDGROUP外に置くようにするほうが望ましい．これは，ライブラリを利用する側のプログラムのコードやデータの領域を圧迫しなくするためです．

このようにすれば，スモールモデルからでもラージモデルからでも，同じ関数を呼び出して使うことが可能になります．実際，Microsoft CやTurbo Cに添付されているグラフィックライブラリはこのように作られています．

メモリモデルのカスタマイズ (Customizing memory models)
標準以外のメモリモデルの作成

カスタマイズというのは，コードとデータそれぞれに対してアドレスサイズを指定したり，スタックの配置を規定したりして，標準以外のメモリモデルを作成する機能です．Microsoft Cでは，メモリモデルは通常/AT,/AS,/AM,/AC,/AL,/AHの各オプションスイッチのどれかで指定しますが，これら以外に図12 (1) のような小文字による指定方法があります．すなわち，通常のメモリモデル指定は，図12(2)のような指定と等価なわけです．

したがって，たとえば/Ashdと指定すれば，いわゆるヒュージコンパクトモデルになります．また，スタックの設定を変更することにより，スタックをDGROUPから追い出すことも可能です．もっとも，これを行うには，スタートアップルーチンを修正してライブラリを作りなおす必要があります．それに，標準ライブラリの関数はDS==SSを前提にしていますので，その場合標準ライブラリ関数はいっさい使えなくなります．つまり，単純に/Auや/Awを指定すれば大きなスタックを確保できるというわけではありません．

Microsoft C V.6.0では，スタートアップルーチンを修正して独立したスタックを設定する方法がREADME.DOCに記述されていますから，おそらくコンパクト／ラージモデルの標準ライブラリはすでにDS != SSで作成されているものと思われます．次のバージョンでは正式にサポートされるのかもしれません．

このカスタマイズ機能は，どちらかというと，MS-WINDOWSやOS/2でダイナミックリンクを実現するための機能であるといえます．ダイナミックリンクライブラリのコードは，DS != SSを仮定して作成する必要があるからです．MS-DOS上では，デバイスドライバかTSRプログラムでも作る場合以外はあまり必要はないでしょう．

なお，Turbo Cにはこういったカスタマイズの機能はありませんでしたが，Turbo C++ではこれに相当する機能の一部がサポートされています．-mt!, -ms!, -mm!オプションがそれで，これらはそれぞれタイニィ／スモール／ミディアムモデルでDS != SSを仮定するものです．これも，近い将来のWINDOWS／OS/2対応への布石と思われます．

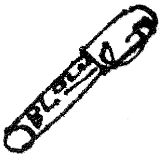


図12 Microsoft Cメモリモデルの指定

```
(1)  /As      コードアドレス near
     /Al      コードアドレス far
     /An      データアドレス near
     /Af      データアドレス far
     /Ah      データアドレス huge
     /Ad      DS == SS を仮定する.
     /Au      DS != SS を仮定し，各関数のエントリでDSにDGROUPの値を設定する.
     /Aw      DS != SS を仮定するが，各関数のエントリでDSに値を設定しない.

(2)  /AT, /AS     /Asnd
     /AM          /Alnd
     /AC          /Asfd
     /AL          /Alfd
     /AH          /Alhd
```

3 オーバーレイ (Overlay)

メモリモデルの話をしたついでに，オーバーレイについても触れておきます．オーバーレイとは，同一のメモリ領域に複数のプログラムコードを配置し，入れ替えながら使用する機能です．MS-DOS上でアプリケーションが使用できるメモリ空間はせいぜい500KB程度に限られていますから，大規模なプログラムではオーバーレイの技法はよく使用されます．

オーバーレイの対象になるのはあくまでプログラムコードだけです．データ部分に関してはいっさい対象になりません．入れ替えの必要が生じると，自動的にディスク上から該当するモジュールがロードされます．

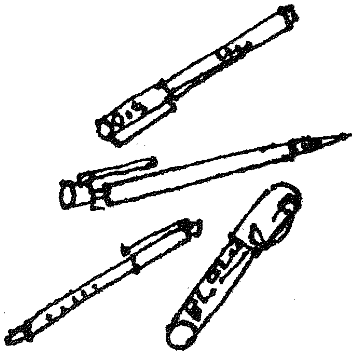
Microsoft Cでは，V.3.0のときからオーバーレイ機能を標準でサポートしてきています．Turbo Cでは，V.2.0まではオーバーレイ機能はサポートしていませんでした．Turbo C++になってはじめて，VROOMMという名前でサポートしています．

Microsoft Cのオーバーレイ機能
プログラムコードをセグメント単位で入れ替え

Microsoft Cのオーバーレイ機能は，プログラムコードをセグメント単位で入れ替えます．すなわち，オーバーレイ部分とするモジュールは，たがいに異なるセグメントをもっていなくてはならず，基本的にはミディアムモデルやラージモデルのような複数のコードセグメントをもつメモリモデルでないと使用できません．オーバーレイ部分のコードは，DGROUPのすぐ下(低位アドレス)のメモリ領域に配置されます．この領域の大きさは，もっとも大きなオーバレイモジュールの大きさとなります．

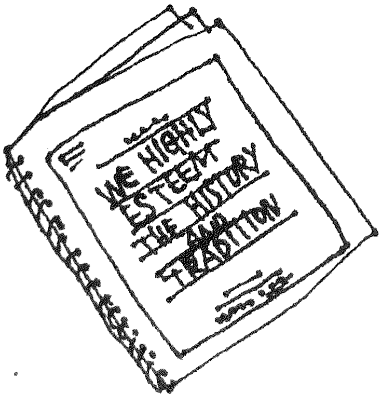
オーバーレイモジュールの指定は，CLまたはLINKのコマンドラインで，オーバーレイ化するモジュール名を括弧で囲んで記述することで行います．

このオーバーレイ機能は，V.3.0のときからサポートされていましたが，V.6.0になってもまったく変更されていません．まったく機能強化されていないため，現時点ではTurbo C++のVROOMMに比べると，EMS対応などの機能面で少々見劣りするものになってしまっています．



Turbo C++のオーバーレイ機能 (VROOMM)
不要なモジュールをEMSメモリ，プロテクトモードメモリへ退避

VROOMMとは，Virtual Runtime Object-Oriented Memory Managerの略です．名前からしてなにか非常に高度なものという印象を受けますが，使う側にとってはそんなにむずかしいものではないようです．基本的に，Microsoft Cのオーバーレイ機能と同様にプログラムコードをセグメント単位で入れ替えるものです．したがって，やはりミディアムモデルやラージモデルのような複数のコードセグメントをもつメモリモデルでないと使用できません．オーバーレイ部分のコードは，スタックのすぐ上（高位アドレス）のメモリ領域に配置されます．この領域の大きさは，_ovrbufferという外部変数を初期化することで変更できます．デフォルトではもっとも大きなオーバレイモジュールの大きさの2倍となります．



ワンラインNEWS
NTTの使用料回収代行サービス「ダイヤルQ2」を利用した商用BBSが登場．業者側は使用料回収が容易になり，ユーザー側は加入手続きが簡単になるメリットがある．

オーバーレイモジュールの指定は，TCCのコマンドラインでは-Yオプションおよび-Yoオプションで行います．

VROOMMはオーバーレイモジュールを入れ替える際に，不要となったモジュールをEMSメモリやプロテクトモードメモリへ退避する機能ももっています．この機能を使用すると，モジュールをロードするときに2回目以降はディスクへのアクセスがなくなりますので，プログラムの実行の高速化が期待できます．しかし，プログラムコードを直接EMSメモリやプロテクトモードメモリ上で実行するわけではありません．なお，EMSメモリやプロテクトモードメモリへの退避の機能を使用する場合は，mainの先頭部分で_OvrInitEms関数，_OvrInitExt関数をそれぞれ呼び出さなくてはなりません．

Microsoft CでもTurbo C++でも，オーバーレイ部分の関数への呼出しは，内部的にINT 3Fh命令に変換されて，オーバーレイマネージャによって間接的に処理されます．したがって，あるオーバレイモジュール内の関数から，ほかのオーバーレイモジュールの関数を呼び出すことも可能です．しかしこれはモジュールの入れ替えを伴いますから，性能を落さないためには，なるべく避けるようにするべきです．

リスト3にオーバーレイの非常に簡単な例を示します．この場合，o1, o2, o3が同じ領域で入れかわりながら実行されます．なお，オーバーレイモジュールは，1つの実行形式ファイルの中にまとめて入ってしまう点に注意してください．

リスト3　オーバーレイプログラムの作成例

```
/* main.c */
void Overlay1(void);
void Overlay3(void);

main()
{
    printf("in main¥n");
    Overlay1();
    Overlay3();
}


/* o1.c */
void Overlay2(void);

void Overlay1(void)
{
    printf("in Overlay 1¥n");
    Overlay2();
}


/* o2.c */
void Overlay2(void)
{
    printf("in Overlay 2¥n");
}


/* o3.c */
void Overlay3(void)
{
    printf("in Overlay 3¥n");
}
```

Microsoft C の場合，次のようにしてコンパイル・リンクします．

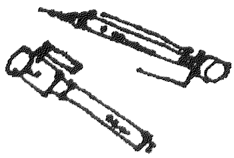
```
cl /AL main.c (o1.c) (o2.c) (o3.c)
```

Turbo C++ の場合，次のようにしてコンパイル・リンクします．

```
tcc -ml -Y main.c -Yo o1.c o2.c o3.c
```

今回はメモリモデルとオーバーレイの話をしました．上級者の方なら，Microsoft CまたはTurbo Cの少なくともどちらか片方については，このレベルの知識は把握していることと思います．実際，こういう細かいレベルの知識は，自分の使う処理系についてだけ把握していれば実用上は十分なのです．しかし，こういった比較を行っていくと両処理系の設計思想の違いといったものが見えてきてけっこうおもしろいと思っているのですが，いかがでしょうか？

次回は，ポインタやインラインアセンブラなどの話をしたいと思っています．懲りずにつき合っていただければ幸いです．



98活用アセンブラ技法

ビデオシステム編 第2回

半角外字フォントの読み込みと設定

星野 操

今回は，前回説明した構造化マクロを利用した実例を示します．最初のプログラムは，外字フォントの読み込みプログラムです．

前回のマクロの修正

@whileと@untilで複合条件をサポート

前回は宿題扱いにしましたが，今回のプログラムのつごう上 (:-) @untilにも複合条件が使用できたほうが便利なので，@whileと@untilで複合条件をサポートしてみました．

リスト1に，これらの新しい内容を示します．前回のcondx.incの@while/@untilをこれらと入れ替えて使用すると考えてください．

変更は，いずれも最初の部分に条件が空である場合の処理を付加しただけです．条件が成立したときのエントリー，不成立のときのエントリー，そのまま流れてきたときのエントリー，ループのジャンプ先ラベルを同時に管理しなければいけないので，うまく管理順番を並べないと，めんどうなことになります．必要に応じて，$tempをラベル名の退避用に使用しています．

リスト1

```
@until  macro   cond
        $xlist
        ifb     <cond>
                $pop    MC_PTR
                $set    $temp,$1%$val   ;; remember failure entry point label
                $setn   $val,$x%$stack  ;; success entry point for @or
                $label  $1%$val
                $pop    BEGIN_PTR
                $jcond  mp,<short $1%($val-1)>  ;; goto exit label
                $label  %$temp          ;; failure entry point
                $jcond  mp,$1%$val
                $label  $1%($val-1)     ;; and define loop end
                $list
                exitm
        endif

        $pop    BEGIN_PTR               ;; normal case
        $set    $temp,$1%$val
        if      ($+2-$temp) gt 127      ;; and calc jump range
                $jcond  cond,$+5        ;; long range
                $jcond  mp,$1%$val
        else
                $jncond cond,$1%$val    ;; short range, direct jump
        endif
        $label  $1%($val-1)             ;; and define loop end
        $list
        endm

@while  macro   cond,range
        $xlist
        ifb     <cond>
                $pop    MC_PTR
                $jcond  mp,$+5          ;; jump to sccess entry point
                $label  $1%$val         ;; failure entry point
                $setn   $val,$x%$stack  ;; remember sccess entry point
                $set    $temp,$1%$val
                $swap   BEGIN_PTR,WHILE_PTR     ;; modify ID
                $jcond  mp,$1%($val-1)  ;; exit loop
                $label  $temp           ;; success entry point
                $list
                exitm
        endif

        $swap   BEGIN_PTR,WHILE_PTR     ;; modify ID
        ifb     <range>
                $jncond cond,$1%($val-1)        ;; short range exit
        else
                $jcond  cond,$+5        ;; long range exit
                $jcond  mp,$1%($val-1)
        endif
```


ワンラインNEWS
キユーピーはマヨネーズなど調味料の販売促進をめざし，料理メニュー提案システムを開発，稼働させた．肉や魚・野菜などの料理がデータベース化されている．

```
        $list
        endm

```

今回のひとつめ, utils.inc

ユーティリティマクロの定義

では，今回のプログラムを示すことにします．最初は，いくつかのユーティリティマクロを定義してあるutils.incです．これは，マクロ定義マクロのサンプルにもなっています．これらのマクロの記述には，condxで定義されているマクロ，あるいは変数などを使用しています．ですからこのファイルを読み込む前に，condx.incを読み込んでおく必要があります．

関数定義マクロ

アセンブラで簡単に関数を定義

まず，最初に定義してあるのは簡単に関数を定義するものです．手続きを作成するためには，アセンブラではproc疑似命令を使用します．新しいmasmではproc疑似命令は引数を取ることができるようになり，自動的にスタックフレームを生成するようになっています．

さらに，si/diの自動退避・復帰コード生成の機能や，ローカル変数の定義もできるようになっています．つまり，高級言語とのインターフェースが非常に容易にできるようになっているのです．

ただ，このようにして作成された関数を「アセンブラから」呼び出す場合には，事前にスタック上に引数をセットアップする必要があります．この場合，どうしてもオーバーヘッドが大きくなります．またコードを書く上でも，毎回レジスタに値を読み込んでからpushするのは手間ですし，引数の不整合を起こす可能性もあります．

本来は，引数をレジスタ渡しにして

リスト2

```
;----------------------------
;ユーティリティマクロ
; utils: other utility macros
;
; $Log: RCS/utils.inc $
; revision 1.2 TAN 90/10/30 05:33:30
; just modified
;
; revision 1.1 TAN 90/10/29 18:45:39
; Initial revision
;
;----------------------------

        .xlist
        .xall

;----------------------------
;関数定義マクロ
;
; FUNCTION DEFINITION MACROS
;
;----------------------------
; function support routine
; for 186 or later cpu only (including v30)
;
; func: create function header, and define function calling macro
; endf: shows end of function
; ext_func: define function calling macro only

        .186

ext_func        macro   nm,arg
        $xlist
        $setn   $val,0                  ;; count number of argument
        irp     x,<arg>
                $inc    $val
        endm
        $setn   $a_&nm,%$val            ;; register it
nm      macro   argl                    ;; function calling macro definition
        $setn   $val,$a_&&nm            ;; get number of declared argument
        irp     x,<argl>
                $list
                push    x
                $xlist
                $dec    $val            ;; and compare it to actual ones
        endm
        if      $val ne 0               ;; argument number differnt,
                .err                    ;; cause error
                $if2
                        $list
                        %out    UTILS Error: argument error on calling nm
                        $xlist
                $endif2
        endif
        $list
        call    _&&nm                   ;; call under-scored function
        $xlist
        endm
        $list
        endm

func    macro   nm,arg,opt
        $xlist
        ext_func        nm,<arg>        ;; define schema of function
        ifb     <opt>
_&nm    proc    arg                     ;; and procedure declaration
        else
_&nm    proc    uses opt, arg
        endif
$fname  equ     <_&nm>                  ;; reserve function name for later use
        $list
        endm

$endf   macro   name                    ;; local macro to use argument evaluation
        $xlist
name    endp
        $list
        endm

endf    macro                           ;; shows end of function
        $xlist
        $endf   %$fname
        $list
        endm

;----------------------------
```

ワンラインNEWS　米モトローラ社は32ビットMPU 68040を量産開始した．68040は1989年に発表されたが，設計の欠陥や生産設備の問題から半年以上量産が遅れていた．

最適化したほうがよいに決まっています．実際，MS-Cもいちばん新しいバージョンではレジスタ渡し関数がサポートされたと思いました．ただし，これらを人間が管理するのはかなりむずかしいでしょう．ですから多少の効率低下はあるにしても，引数はスタック渡しにしておくにこしたことはありません．最終的にスピードに問題が生じた時点で最適化すればよいのです．

効率には目をつぶったのですから，その分書きやすくなっていなければなりません．構造化マクロと同じことです．ですから，ここでは引数の記述を簡略化できるようなマクロを定義してあるのです．

func／endf

関数定義には，マクロfuncとendfを使用します．関数定義は，図1のような書式で記述します．

図1で，引数リストはカンマで区切った引数を<>で括ったものです．個々の引数の中にはスペースが含まれていてもかまいません．また，関数の中でsiレジスタやdiレジスタを使用する場合には，引数リストのあとに使用するレジスタ名siまたはdiを<>で括って記述します．

マクロfuncは，次のような処理をします．まず，関数名と同じ名前のマクロを定義します．ここで定義されたマクロは，引数リストと同じ構造をもった引数を1つ取り，それを順番にスタックにプッシュした後，関数名の先頭にアンダースコアをつけた手続きを呼び出すようなコードを生成します．

図1

```
func  関数名  <引数リスト> [<si di>]
      命令. .
ednf
```

```
;セットアップマクロ
;
; PROGRAM SETUP MACRO
;
;----------------------------
; this macro setup registers and argc/argv arguments
; used in startup routine.
; this macro setup arguments in a stack. so subroutine
; is not suitable for the purpose.

        .data
$psp    dw      ?                       ; to remember psp

_argc   equ     <word ptr [bp]>         ; argument definition
_argv   equ     <word ptr [bp+2]>

        .code
@setup  macro
        $xlist
        mov     bx,ds                   ;; save psp segment value
        mov     ax,_DATA                ;; set data segment value
        mov     ds,ax
        mov     $psp,bx                 ;; and set psp value

        mov     bx,ss                   ;; adjust stack segment
        sub     bx,ax                   ;; calculate paragraph offset
        add     bx,bx                   ;; and mul 16
        add     bx,bx
        add     bx,bx
        add     bx,bx
        mov     ss,ax                   ;; correct ss/sp
        add     sp,bx

        sub     sp,128+130+4            ;; make work space

        mov     bp,sp                   ;; create frame pointer
        lea     bx,[bp+4]               ;; array top address
        mov     _argc,1
        mov     _argv,bx
        mov     word ptr [bx],0         ;; program name
        inc     bx
        inc     bx

        push    ds
        mov     ds,$psp                 ;; copy argument
        mov     si,80h
        push    ss
        pop     es
        lea     di,[bp+130+4]           ;; argument buffer address
        mov     cx,80h/2
        cld
        rep     movsw
        pop     ds

        lea     si,[bp+130+4+1]         ;; si:argument string top
        @begin
                @begin                  ;; skip space/tab
                        lodsb
                        @mc
                                cmp     al,' '
                                @cond   nz
                                cmp     al,9
                                @cond   nz
                @until

                cmp     al,0dh          ;; check end of argument
        @while  nz
                dec     si              ;; save top of string
                mov     [bx],si
                inc     si
                inc     bx              ;; advance pointer
                inc     bx

                @begin                  ;; find end of string
                        lodsb
                        @mc
                                cmp     al,' '
                                @cond   z
                        @or
                                cmp     al,9
                                @cond   z
                        @or
                                cmp     al,0dh
                                @cond   z
                @until

                mov     byte ptr [si-1],0       ;; set terminate char
                inc     _argc                   ;; increment argc

                cmp     al,0dh          ;; post check
                @if     z
                        @break
                @endif
        @repeat

        mov     word ptr [bx],0         ;; terminate pointer array
```


ワンラインNEWS
コンテックは日本電気のPC-9800シリーズ用のRAMボードを発売した．同ボードはEMSに準拠した製品で従来の製品に比べ設定のしやすさに重点が置かれている．

ここで，このようなマクロ作成に先立って引数の個数が数えられ，$a_を関数名に前置した数値マクロにそれを設定します.「関数名」をもったマクロは，実行時に与えられた引数の数とこの数値マクロを参照することで，正しい個数の引数が与えられたかどうかを判別します．その後関数名を$fnameに記録し，最後に関数名に_を付けた名前のラベルでproc疑似命令を実行します．これによって，手続き本体の宣言がなされるのです. si/diの使用指示がある場合には，ここでusesが宣言されます．

$fnameに記録された関数名は，あとでendfマクロの中で使用されます. proc疑似命令とendpで手続きを括る場合には, endpで再び関数名を指定する必要があります．この$fnameを使用することによって，endfで関数名を再度記述する手間を省いているわけです．

このような呼出し形式を用いているわけですから，ほかのモジュールでも同様に関数呼出しマクロを定義できるようにしてやらないといけません．マクロext_funcはそのために用意されています. funcのかわりに，ext_funcを使用すれば，関数呼出しを実行するマクロのみが定義されます．funcも内部でext_funcを呼び出しています．

func(ext_func)のようなマクロは，そのマクロが実行されたときに別のマクロを作成するようになっています．マクロの定義部分にマクロ定義形式を記述するだけで，このようなマクロ定義マクロを作成することが可能になるのです．このようなことができるmasmの仕様は，非常に便利であるといえます．

たとえば, CPPの#defineでは, 別の文字マクロを定義するマクロを記述で

```

        $list
        endm

;----------------------------
;文字列定義マクロ
;
; STRING DEFINITION MACRO
;
;----------------------------
; use to define null terminated string constants, output string.
; @string define string in a data segment, and load its offset value
; into dx or other assumed register.
; @puts output argument null terminated string to stdout.

@string macro   mes,reg
        local   l                           ;; string address
        $xlist

        .data
        $list
l       db      mes                         ;; define string
        db      0                           ;; additional zero
        $xlist
        .xcref  l

        .code
        ifb     <reg>
                $list
                mov     dx,offset l     ;; default, set address into dx
                $xlist
        else
                $list
                mov     reg,offset l    ;; register name assumed case
                $xlist
        endif
        $list
        endm

@puts   macro   mes
        $xlist
        mov     bx,mes                      ;; get offset of string
        @begin
                mov     dl,[bx]             ;; take one char
                inc     bx
                or      dl,dl               ;; upto terminator
        @while  nz
                dos     <2,,>               ;; use dos-1 function
        @repeat
        $list
        endm

;----------------------------
;DOSアクセスマクロ
;
; DOS ACCESS MACRO
;
;----------------------------
; use to invoke dos.
;
; first, create register alias
; $dos_reg_0 is ah
; $dos_reg_1 is bx
; $dos_reg_2 is cx
; $dos_reg_3 is dx
; they are used in macro 'dos'

        $setn   $val,0
        irp     x,<ah,bx,cx,dx>                 ;; more register can be added
                $set    $dos_reg_%$val,x        ;; create register alias
                $inc    $val                    ;; increment register number
        endm

;----------------------------
;DOS呼出しマクロ
; dos: dos calling macro
; use as following:
;       dos     <ah,bx,cx,dx>
; arguments must be constants, or memory/register.
; they may be blank. in the case, values not loaded into register

dos     macro   arg
        $xlist
        $setn   $val,0                          ;; $val shows register number
        irp     x,<arg>
                ifnb    <x>                     ;; blank: skip setting value
                        $set    $temp,$dos_reg_%$val    ;; create register name
                        ifdifi  <x>,<$temp>     ;; check duplicate load
                                $list
                                mov     $temp,x ;; set value into register
                                $xlist
                        endif
                endif
                $inc    $val                    ;; advance register pointer
        endm
```

きませんから，不便を感じることが多々あります．

関数定義マクロの制限

さて，このような関数定義のマクロですが，都合上いくつかの制限があります．まず第一に，引数はすべてword形式でなければいけません．もしも，dword形式の引数を取りたい場合には，それらを演算子shlなどで処理して，２つのword引数に分割する必要があります．受け側でも同様です．

次に，引数の数を数えていることから，可変個の引数を取ることはできません．引数の評価は左から右に行われていることも理由の１つとなります．つまり，引数の取り方はC形式ではなくPASCAL形式になっているのです．ですから，このライブラリを使用する場合にはあらかじめ .mode疑似命令の第２引数でpascal書式を指定しておく必要があります．

pascalモードになっていれば，スタック調整の機械語コードはret命令によって自動的に生成されるので，ユーザープログラムのほうで注意する必要はありません．C形式で書けるようにした場合には，呼出し側のほうでスタックの調整が必要となるので，マクロが作りにくくなります．

引数の受け渡しはPASCAL形式なのですが，関数名に_が付加されるのはCと同様です．ですから，高級言語とのインターフェースを取る場合には，C言語で使用するときにはpascal宣言子を明示する必要がありますし，また関数名にも_をつけてやる必要があります．

引数のスタックへのプッシュは，直接push命令が使用されています．ですから，引数に定数が使用された場合に問題が生じる場合があります．80286

```
        $list
        int     21h                             ;; then call dos
        endm



        .list

```

以降のCPU，またV30なら定数のpushはサポートされています．しかし，8086ではこれはサポートされていません．

PC-9801のVM以降にシステムを限定するなら問題はないのですが，8086の上で動かすプログラムにこのマクロを使用する場合には，なんらかの措置（たとえば，typeで引数の型を調べて，定数なら一度レジスタに読み込んでからpushするなど）をほどこす必要があるでしょう．

@setup

次に定義してあるマクロ@setupは，プログラムのセットアップをするマクロです．セグメントのセットアップをし，引数を解析することによってCと同様にargc/argvを作成します．引数文字列やargv配列はスタックの中に取るようになっています．このマクロは，プログラムの起動時に１回しか使用されませんが，スタックを操作するために関数ではなくマクロにしてあります．

最初に，pspの記憶領域と_argc/_argvの変数宣言があります．ここでは文字列マクロで定義を行っていますが，proc疑似命令の引数の場合も，同様な文字列マクロが自動的に定義されているだけです．

このマクロは，最初にDSのセットアップとpspの保存を行います．さらに，スタックセグメントがデータセグメントと同一の値になるようにSS/SPの調整を行います．

その後，スタックフレームの生成とローカル変数のための領域を確保します．引数のコピー128バイトと，そのためのポインタ配列領域，そして_argc/_argv変数用の領域として４バイトを確保します．それから，pspの中から引数をスタックの中にコピーし，これを解析します．そして，文字列の先頭を配列の中に格納し，_argcの値を調節します．

この解析は引数の最後にCRが現れるまで行われ，最後にnullを配列に格納します．

ここに出てくる２つの@begin-@untilループの中で，多重条件分岐を使用しています．たとえば，最初のループではスペースかタブ以外の文字が現れるまでループを回るようになっています．@mcのあとの２つの条件，@cond　nzが双方満たされればループを脱出するわけです．連続した条件，つまりand条件はこのように@condを連続して使用します．

また，２番目のループではスペース，タブ，CR以外の文字である間ループを回します．@orはこの例のように使用します．つまり，@orで区切られたブロックのどこかの@condが全部満たされれば，条件は真になり，ここではループを脱出することになります．このように@mcを使用すれば，１段階の積和条件分岐を記述できるのです．

文字列定義／出力マクロ

引数にとった文字列の収納と出力

次に，文字列定義マクロと文字列出

ワンラインNEWS
三菱電機は半導体の主要製品である1MビットDRAMの生産をピーク時に比べ20%減産した．これは米国市場の低迷感と，4MビットDRAMに生産体制をシフトするため．

力マクロが定義されています．.DOSの文字列出力には$で区切られた部分までを出力するファンクションがありますが，$を区切り文字にするとファイル名などの文字定数を定義したいときに不便です．

@string

ここで定義されている@stringは引数に取った文字列を「データセグメントに」定義し，そのオフセットをDXまたは指定したレジスタに格納するようになっています．これによって，.dataでセグメントを明示的に切り替えることなしに，プログラム中に文字列定数を記述できます．また，文字列に無用なラベルを付加する必要もありません．このマクロで作成された文字列は，0で区切られていますから次の@putsマクロのほかに，DOSの各種文字列引数として使用することができます．

@puts

次の@putsは，引数に取った文字列を出力する命令です．DOSのファンクション2を使用し，文字列を分解して出力するようになっています．ここで引数に取れるのは，レジスタ・変数・アドレス定数などです．ローカル変数のシンボルは，アドレスではないので一度LEA命令でレジスタにアドレスを設定してから渡すようにします．前記@stringを使用した場合にはDXを，そのほかのラベル付き文字列定数であればoffset付きのラベル名を使用すればよいでしょう．

offset付きのラベル名を引数にする場合は，全体を<>で括ることをわすれないようにします．普通の命令ではスペースはトークンの区切の意味しか持ちません．しかし，マクロの引数になる場合は，スペースは引数の区切として解釈されてしまうからです．

このマクロのようなものは，プログラム中で多く使用するようであれば関数に書き直したほうがよいでしょう．

dos

最後に定義してあるのは，dosというマクロです．これは，<>で括った引数を1つ持ち，その値をレジスタに設定してint21を発行するというものです．このときに使用される文字列マクロが$dos_reg_?というマクロで，これはdosマクロの定義の直前に定義されています．

基本的にこれらにah～dxの値を設定するわけで，equ疑似命令を4つ使えばよいだけです．ただ，今後ESなどのレジスタを使用したい場合があるかもしれません．このため，簡単に拡張できるようにirpでループを回して定義することにしました．ここでは，$setnや$set／$incなどをCONDXの一部のマクロを流用して目的を達しています．

dosマクロは，<>の中に4つまでの引数を取り，それぞれをレジスタに代入します．ただし，内容がブランクであれば値の設定はなされませんし，同一のレジスタ名を引数に取った場合でも無駄なコピーはしないようになっています．

<>の中の引数をアクセスする場合には，irpで分解するわけですが，このときにループカウンタを併用し，対応するレジスタ名を$setマクロで$tempに代入します．そして，それをあとの命令コードで使用するわけです．

引数は，ah（DOSの機能コード）／bx/cx/dxの順になっています．alに値を設定することが必要な場合は，あらかじめ（dosマクロを使用する前に）値を読み込んでおきます．もちろん，簡単にアクセスできないオブジェクトをレジスタに設定する必要がある場合には，事前に値を読み込み，dosマクロでの値の設定をオミットすればよいのです．

今回のふたつめ，dosfile.inc

ファイルのオープン，リード，ライトをサポート

もう1つ，今回のプログラムで使用するインクルードファイルdosfileについて触れることにしましょう．

今回のプログラムでは，ファイルアクセスが必要になります．ファイルのオープンや基本的な読み書きアクセスをサポートしたのがこのインクルードファイルです．これらの機能は，関数で定義してライブラリとしてリンクしたほうが方針としてはよいと思います．ただ，アセンブラからのみ使用することを考えると，実際にはレジスタに直接値をセットしてint21を発行するだけで十分でしょう．

そのような場合に，スタック経由のパラメータ渡しをするのもつまらない話です．コードサイズ自体も大きくなります．なにより，ライブラリにすると別個の管理が必要になります．ですから，ここではこれらの機能は，みなマクロで書いてしまいました．もちろん，関数に書きなおすのは簡単です．必要があればライブラリの体裁を整えるとよいでしょう．

このマクロでは，dosマクロを使用しています．ですから，utils.inc（ひいてはcondx.inc）を併用することが必要です．

create/open/close/read/write

まず，最初にCと同じような名前の

定数の定義があります．これらは，DOSパラメータとして使用するものです．O_xxxxという定数はopenを呼び出すときに使用し，FA_xxxxという定数はcreateを呼び出すときに使用します．

用意されているマクロは，ファイルアクセスに必要最低限なcreate/open/close/read/writeのみです．もちろん，必要に応じてtellなどのマクロを作成することも簡単です．

これらのマクロでは，通常のマクロ引数を取ることになっていますから，引数全体を<>で括る必要はありません．そのかわり，空白が必要な場合(たとえば，offset　stringのような形式の引数が要求される場合）には，その引数自体を<>で括らなければいけません．これは，@putsと同様です．

引数の順番は，Cの対応する関数とほぼあわせてあります．たとえば，read/writeでは，まずハンドル，バッファアドレス，長さの順になります．帰り値は，基本的にDOSコールした結果そのままです．フラグも操作していないので，必要に応じてキャリーフラグのチェックをしなければいけません．

さて，create/openはともにファイル名を引数に取ります．第2引数としてそれぞれのモードを指定することもできますが，これらがオミットされた場合にはcreateではノーマルファイル，openではread onlyが仮定されます．closeでは，ハンドルを引数に取るだけです．これらのマクロでは，とうぜん内部で使用されるレジスタは破壊されることになります．

read/writeも同様にdosマクロを利用してレジスタに対する値の設定，それからint21の起動を行っているだ

リスト3

```
;-----------------------------
;ファイルアクセスのサポート
; dosfile: dos file system support macros
;
; $Log: RCS/dosfile.inc $
; revision 1.2 TAN 90/10/30 05:33:18
; just modified
;
; revision 1.1 TAN 90/10/29 18:45:47
; Initial revision
;
;-----------------------------

        .xlist
        .xall

;-----------------------------
;オープンモード定数
; open mode constants

O_RDONLY        equ     0       ; for read only
O_WRONLY        equ     1       ; for write only
O_RDWR          equ     2       ; for read/write

;-----------------------------
;ファイル属性定数
; file attribute constants

FA_RDONLY       equ     1
FA_HIDDEN       equ     2
FA_SYSTEM       equ     4
FA_LABEL        equ     8
FA_DIREC        equ     10h
FA_ARCH         equ     20h

;-----------------------------
;ファイル作成マクロ
; create: file create macro
; argument---
;       fname: pointer to file name
;       attrib: file attribute. default: normal
; return---
;       carry, ax: handle or error code
; broken---
;       cx,dx

create  macro   fname,attrib
        ifb     <attrib>
                mov     cx,0
        else
                mov     cx,attrib
        endif
        dos     <3ch,,,fname>
        endm

;-----------------------------
;ファイルオープンマクロ
; open: file open macro
; argument---
;       fname: pointer to file name
;       access: access mode. default: read only.
; return---
;       carry, ax: handle or error code
; broken---
;       dx

open    macro   fname,access
        ifb     <access>                        ;; set access mode
                mov     al,O_RDONLY             ;; default: read_only
        else
                mov     al,byte ptr access      ;; otherwise, set actual value
        endif
        dos     <3dh,,,fname>                   ;; and call dos
        endm

;-----------------------------
;ファイルクローズマクロ
; close: file close macro
; argument---
;       handle: file handle
; return---
;       carry, ax: handle or error code
; broken---
;       bx

close   macro   handle
        dos     <3eh,handle>
        endm

;-----------------------------
;ファイルリードマクロ
; read: file read macro
```


ワンラインNEWS
ランドコンピュータは操作者の理解度を判断して，レベルに応じたプログラムが進行するCAI教材開発支援ソフトを1月に発売する．

けです．ただし，引数にレジスタを取ってコンフリクトを起こさないように気をつけます．たとえば，bxにファイル名のアドレスを代入し，これを第3引数に取ると，最初のbxレジスタに対する値のロードの時点で内容が壊されてしまうからです．

本連載ではこれ以上のファイルアクセスを行うことはまずないので，このインクルードファイルの充実をはかることはないでしょうが，いずれにしろ簡単な処理でサポートする機能を増やすことは可能だと思います．

最初のプログラム例，gaiji.asm

半角表示機能を利用して，半角外字フォントを読み込む

さて，以上のようなインクルードファイルを用意してから，アセンブラのプログラム作成に入ります．

簡単な例として，今回はまず外字フォントへの読み込みプログラムをあげてみました．もちろん，condx/utils/dosfileの3つのマクロライブラリを内部でインクルードしてあります．このプログラムは，PC-9801の外字の半角表示機能を利用して，半角外字フォントを読み込み設定するものです．

最初に，簡略化されたセグメント指定を利用するために，dosseg疑似命令が使用されています．それから，拡張モデル指定の .model疑似命令で，small/pascalモードに指定されています．funcマクロのところで述べたように，ここではpascal形式のパラメータ渡しを行っているからです．

データ領域には，ファイルアクセスのための16バイトのバッファと，bios引数のためのデータ領域を用意してあ

```
; argument---
;       handle: file handle
;       buf: pointer to buffer
;       len: requested bytes to read
; return---
;       carry, ax: readed byte or error code
;       buffer: readed data
; broken---
;       bx, cx, dx

read    macro   handle,buf,len
        dos     <3fh,handle,len,buf>
        endm

;---------------------------
;ファイルライトマクロ
; write: file read macro
; argument---
;       handle: file handle
;       buf: pointer to buffer
;       len: requested bytes to write
; return---
;       carry, ax: written byte or error code
;       buffer: written data
; broken---
;       bx, cx, dx

write   macro   handle,buf,len
        dos     <40h,handle,len,buf>
        endm

        .list
```

リスト4

```
;---------------------------
;半角外字読み込みプログラム
; gaiji: load border char data file
;
; $Log: RCS/gaiji.asm $
; revision 1.1 TAN 90/10/29 18:46:32
; Initial revision
;
;---------------------------

        dosseg
        .model  small,pascal

        include condx.inc
        include utils.inc
        include dosfile.inc

        .stack  2000h

        .data
buf     db      16 dup (?)              ; file buffer area
gwork   dw      ?                       ; bios arguments
gbuf    dw      16 dup (?)

;---------------------------
;イニシャライズルーチン
; initialize routine

        .code
entry:
        @setup
        push    _argc
        push    _argv
        call    _main
exit:
        dos     <4ch,,,>

;---------------------------
;メインルーチン
; main routine

func    main    <argc, argv>
        local   fname, fd

        ; open file
        push    ax
        pop     ax
        @string <"bdfont.dat">          ; data file name
        mov     fname,dx                ; reserve it
        open    dx                      ; and try to open
        @if     c
                @string <'file open error: '>
                @puts   dx              ; output error mesage
                @puts   fname
                @string <0dh,0ah>
                @puts   dx
                mov     al,1            ; return value
```

ります．入力ファイルのフォーマットは，上位rowから順に16バイト（半角1キャラクタ分）のデータを順に書き込んであるだけです．それを，拡張して外字フォントに読み込みます．

PC-9801のハードウェア

PC-9801の漢字アクセスハードウェアは，1半角キャラクタに対して2バイトがわり当てられています．漢字1文字を表示するために，4バイトわり当てられているわけです．ところが，実際には(少なくともVM以降では)左側のコードが漢字コードであれば（ワードアクセスして上位バイトが0以外であれば)，右側のコードは無視され，必ず全角の文字が表示されます．

一応，上位ビットに左右を示すフラグビットというのが用意されていますが，これはハードウェアからはほとんど無視されています．左側の2バイト分のコードに右側だということを示すフラグをセットしても，漢字の右半分が表示されることはないのです．

ただ例外として，外字だけは左右を別個に表示させることが可能です．左側の2バイトが外字コードであり，その次に別の外字キャラクタや半角キャラクタが記述されていれば，外字の左半分（フラグによっては右半分）だけが表示されるのです．この機能を利用すれば，半角外字が定義でき，それによってPC-9801に欠けているダブルボーダーなどのキャラクタを表示することが可能となります．実際，この機能を利用して半角バックスラッシュを表示するような常駐プログラムがいくつかあります．

半角外字の制限

ただし，この半角外字にはいくつかの制限があるので注意しなければいけ

```
                jmp     exit
        @endif
        mov     fd,ax                   ; save file handle

; read
        @do     si,64,0                 ; for 64 characters
                read    fd,<offset buf>,16      ; read file
                @do     bx,16,0
                        mov     di,bx   ; adjust for bios argument
                        add     di,di
                        mov     al,buf[bx]
                        mov     ah,al
                        mov     gbuf[di],ax
                @loop

                mov     ax,1a00h        ; prepare bios arguments
                mov     bx,ds
                mov     cx,offset gwork
                mov     dx,si           ; calculate jis_code
                shr     dx,1
                adc     dh,76h          ; range 7600~761f, 7700~771f
                int     18h             ; invoke bios
        @loop
        mov     al,0
        ret
endf

        end     entry

```

ません．まず，半角外字の上にカーソルがくると，全角幅になってしまうことです．これは，ハードウェアの制限ですからいかんともしがたいものがあります．

次に，異なる外字キャラクタが連続している場合はよいのですが，同じ外字キャラクタが連続している場合に限って，左右フラグが認識されず，つねに左・右の順で外字1文字分が表示されてしまいます．このため，現実問題として外字の左右に別々の半角キャラクタを設定することはできません．万一これらが続いてしまったときに（あるいは，自分自身が2キャラクタ重なったときに）表示が乱れることになるからです．ですから，左右は同じパターンを登録してやらなければいけません．gaiji.asmでファイルバッファが16バイトしかないのは，それを外字の左右に複製してわり当てるようにしてあるからです．

最後に，半角外字のあとに半角漢字(たとえば，半角罫線など)を記述すると，そこがブランクになってしまいます．ですから，半角外字と半角漢字の混用は不可能です．この点については，注意する必要があるかもしれません．

外字ハードウェアについてもうひとつ注意しておくのは，VM以降のハードウェアは，実際には256文字の外字を定義できる，ということです．マニュアルには192文字程度しか定義できないとあります．実際，JISコードで表現すると，それぐらいしかわり当てることはできません．しかしハードウェア上は，不正なJISコードである下位バイトが0~1fhの部分も使用可能です．ですから，DOSの出力などを利用せず画面直接アクセスを前提とすれば，さらに64文字程度の外字領域が使用できるのです．

今回のプログラムでは，ボーダーキャラクタに64文字の領域をわり当てましたから，ちょうどこの範囲に全部納まります．そこで，このgaijiプログラムは64文字をこのエリアの外字フォントに読み込むことにします．

ただし，下位バイトに0を使用する場合そのままだと「半角コード」と同じになるので，半角文字が出力されてしまいます．そのため，上位ビットを立てて80hにしてやる必要がありま

ワンラインNEWS
シャープはニューロ技術利用で，従来の50倍程度の速度で印刷された漢字を識別する方法を開発した．数年内に機械翻訳システムの入力装置として実用化する．

す．今回は，下位バイトに0が当たるキャラクタは無効コードとすることでこの問題を回避しています．

プログラムの流れ

プログラムは，エントリールーチンentryから実行されます．@setupマクロで引数の解析やセグメントのセットアップなどが実行されます．そして，_argcと_argvをスタックにプッシュし_mainを呼び出します．この構造は，ほかのプログラムでも同じになります．

main自身が先にfuncマクロで定義されていればここは，main <_argc, _argv>と記述できるのですが，entryのほうを先に書いているためこの内容を展開してあります．そして，exitラベルのあとでプログラム終了のDOSコールを実行します．このときに，alレジスタに設定されている値がそのままプログラムの帰り値になります．

main自身はfuncマクロを利用して定義されています．pascal形式の呼出しをしていることから，引数を使用しない場合でもargcとargvの引数宣言は欠かすことができません．そのあとに，ローカル変数を2つ用意してあります．このようにするだけで，適切なスタックフレームが自動的に形成されます．

次にファイルを開くわけですが，ファイル名はbdfont.datに固定しています．もちろん，引数をチェックし，ファイル名をコマンドラインで指定することも容易にできます．ここで，1つ問題がありました．TASMの問題ではないかと思うのですがproc疑似命令の直後に.dataでセグメントの切り替えとデータセグメントの中のdb疑似命令があると，その疑似命令に先立ってスタックフレームを生成するコードが付加されてしまうのです．

このコードはもちろんプログラム実行時には実行されませんから，結局暴走状態になります．このため，最初にダミーのpush ax/pop axを付加し，確実にコードセグメントにスタックフレーム生成コードが記述されるようになっています．

ファイル名は@stringマクロで定義しているので，そのアドレスはdxに代入されています．これは一度変数に保存しておきます．その後，dx自身を引数に取りopenマクロを起動します．

エラーが発生すればキャリーが立ちますから，すぐに@ifマクロで状態をチェックすることができます．エラーがあった場合，エラーメッセージとファイル名を出力して終了します．ここでは，終了にはexitに対するjump命令を使用していますが，ret命令を利用してもかまいません．エラーメッセージの出力には，やはり@stringマクロと，@putsマクロを利用しています．

無事にファイルが開ければ，ハンドルを変数fdに代入し，続いて64文字分の外字設定を実行します．ループカウンタには，ほかの関数やマクロ呼出しで破壊されないことが保証されているsiを使用します．繰返しは，@doマクロを利用すれば（多少効率は落ちるのですが）簡単に記述できます．

ループの中では，まずファイル用のデータ領域に16バイト分のデータを読み込みます．真面目にファイルをアクセスするのであれば自前のバッファリングをしないとパフォーマンス上の問題がありますが，ここではたいしたことはないので，DOSのバッファリングに任せてしまいます．

その後，ループを16回まわして，半角分の外字データを左右コピーし全角分のデータに変換します．それから，外字設定のためのbiosを呼び出します．外字フォントのアクセスは，通常はvsync期間しかできません．漢字フォントの表示を停止すれば，常時アクセスできるようになるので高速に外字の設定ができますが，64バイト程度では通常モードでも問題ないでしょう．

引数の一部として外字のJISコードをDXレジスタに設定します．レンジのチェックは行われていないので，7600h〜761fhなどJISコードとしては不正な部分に対しても外字の定義はできます．

64文字分の外字キャラクタは，7600h〜761fhと7700h〜771fhの2つの部分に分解してやらなければいけません．このために，右1ビット分を上位バイトに振ってやることにします．このようにすると，シフトとローテート命令によって簡単に連続した外字コードに復元することができます．

最後に，正常な帰り値を設定して，retすればスタックフレームは自動的に開放され，entryの中に戻り，終了するようになります．

end疑似命令ではentryをエントリー番地として指定してあります．これで外字キャラクタを設定するプログラムは完結します．

このプログラムから実行ファイルを作成するには，tasmとtlinkを利用して普通にコンパイル・リンクすればexeファイルが作成されます．

関数定義マクロや構造化マクロの使用法については，おおむねこのリストでおわかりいただけると思います．

さて，次号ではボーダーキャラクタフォントを作成するプログラムと，それを利用してラインを書くプログラムを紹介します．

Reintroduction to HARDWARES

プログラマのための ハードウェア再入門

鎌田正寿

キャラクタ画面とグラフィック画面 ディスプレイ 2

パソコンでは，いろいろな理由からキャラクタ画面とグラフィック画面の2つを持っているのが普通ですが，最近話題のワークステーションクラスになるとキャラクタ画面はありません．その理由はCPUの処理能力によるものがいちばん大きいでしょう．処理速度の速いCPUを搭載しているワークステーションでは，キャラクタ画面を持たなくても十分な表示速度が得られます．
ではまず，PC-9801を用いてキャラクタ画面で表示する場合と，グラフィック画面に表示する場合のそれぞれで漢字1文字を表示する処理時間を比較してみましょう．

バイト数で比較すると3:128

まず，PC-9801のキャラクタ画面とグラフィック画面について，仕様と表示の操作を見てみましょう．

キャラクタ画面

仕様：漢字1文字分のコード2バイト＋属性1バイトで表示．（40文字×25行）

操作：漢字1文字を出すには合計3バイトのデータをテキストVRAMに書けば自動的にハードウェアが対応するフォントを表示する．

グラフィック画面

仕様：プレーン数は4枚（16色モード時）．（横640ドット×縦400ドット）

操作：まず漢字コードに対応するフォントデータをROMかファイルから取り出す．フォントパターンを表示する色に従ってグラフィックVRAMに書き込む．16ドットの漢字1文字に対してフォントパターンは32バイトになり，プレーン数が4だから合計128バイト分の書き込みが必要．

このように単純にバイト数を比較しても，キャラクタ：グラフィック＝3：128となります．しかも，グラフィック画面の場合はアドレスはドット単位ですからシフトなどの操作も必要となります（図1参照）．

また，スクロールをする際にも，グラフィックの場合にはウィンドウで分割して表示するのが一般的ですので，キャラクタ画面のようにハードウェアでスクロールはできません（ハードウェアでグラフィック画面の一部を移動させるようなものもありますが，かなり複雑なハードウェアが必要となってしまいます）．

さて，文字だけを表示するならキャラクタ画面だけでもできますが，罫線以外の線分を引くとか円を描画するのにはやっぱりグラフィック画面でないとできませんね．そこで，グラフィック画面を操作するのにBIOSやライブラリなどを使わないと，どのくらいの処理が必要になるかを知っていただくためにプログラムを書いてみました（リスト3）．

このプログラムを実行させると，画面上にランダムな線分を表示し続けます．終了するにはどれか任意のキーを押してください（でもリセットキーなんか押さないでくださいね）．グラフィック関係のBIOSは使っていないと書きましたが，表示開始と表示終了だけはBIOSを使っています．これもIN/OUT命令でできるのですが，今回は，PC-9801のハードの説明をするためではなくて，「点を描画する処理」と「線分を描画する処理」の説明をすることが目的だからです．

線分を描画するプログラム

ちょっと話題がソフトよりになってしまいますが処理内容の説明をします．この線分の描画方法は，グラフィック関係の人ならみんな知っていると思いますが，ほかの人はあまり知らないようです．私が最初この方法を知ったときは「うーん，すごい」と感じました．追いかけてみれば別に難しいことはないのですが，「なるほど，やってる人はここまで検討して頑張ってるんだなあ」と……．普通の人だと線分を描画するだけなら何かのライブラリを使うか，単純に浮動小数演算でやっておしまいですものね．

線分の描画方法（Bresenhamのアルゴリズム）

まず最初に$x1<x2$，$y1<y2$で傾きが0から1の間であるような，点(x1, y1)と点(x2, y2)を線分で結ぶことを考えます．

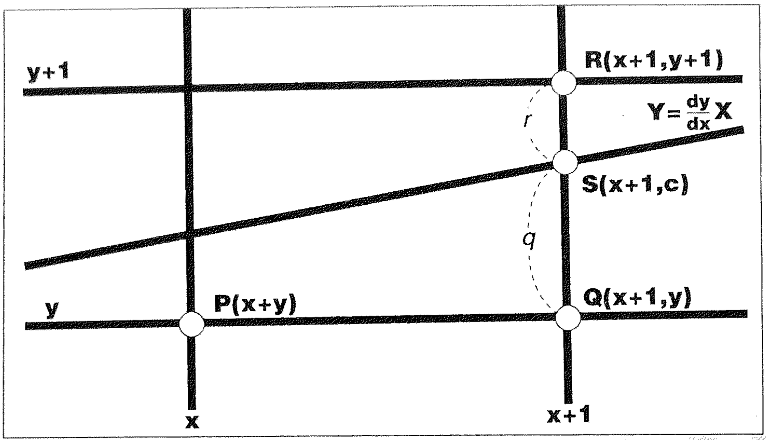
ある点P(x, y)の次の点は，点Q(x+1, y)または点R(x+1, y+1)とな

図1

X →

byte off		byte0								byte1									
bit pos		b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0	b7	…
	Y \ X	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	…
Y	y0	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	…
↓	y1	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	…
	y2	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	…
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

図2■



ります（図2参照）.

この点Qと点Rの間に本来の線分が通過する点S（x+1, c）があり，点Sにより近い点Qまたは点Rを選択すれば最適な線分が引けます.

どちらがより最適かというと距離が近い方ですよね？　ということでそれぞれの距離を計算すると点Qと点Sの距離はc−yで，点Rと点Sの距離は(y+1)−cとなります.

ここで，まずいくつかの値を計算しておきます.

dx=x2−x1
（始点と終点のX座標の差分）
dy=y2−y1
（始点と終点のY座標の差分）

このときの線分の方程式はY=(dy÷dx)×(X−x1)+y1ですが，これを平行移動したY=(dy÷dx)×Xという線分で考えても，判定方法と処理内容は初期座標以外は同じになりますので以後こちらを用います．さて，そこで先ほどの2つの距離を計算します．変数Xがx+1のときのYの値がcですから，c=(dy÷dx)×(x+1)で，点Qと点Sの距離qは

$$q=c-y=(dy\div dx)\times(x+1)-y$$

点Rと点Sの距離rは

$$r=y+1-c=(y+1)-(dy\div dx)\times(x+1)$$

点Pの次が点Qとなる条件はq＜rですのでq−rの計算をすればいいことになります.

$$\begin{aligned}q-r&=(dy\div dx)\times(x+1)-y\\&\quad-(y+1)+(dy\div dx)\times(x+1)\\&=2\times(dy\div dx)\times(x+1)-2\times y-1\end{aligned}$$

ここでdxは差分で必ず正ですから両辺にdxを掛けても条件は同じで，

$$d1=dx\times(q-r)=2\times(dy\times x-dx\times y)+2\times dy-dx \qquad (1)$$

さて，毎回この判定式を計算すると2を掛けている部分はシフトするとしても，乗算が2回も入ってしまいますので，これでは速くはなりません.

そこである点で計算する判定式の結果と次の点で計算する判定式を比べてみます.

この場合，次の点で使う判定式は点Rを選択した場合にはxをx+1で，yをy+1で置き換えた式になりますので

$$d1=2\times(dy\times(x+1)-dx\times(y+1))+2\times dy-dx \qquad (2)$$

d2−d1を計算してみると……

$$d2-d1=2\times(dy-dx) \qquad (3)$$

点Rではなくて点Qを選択した場合には

$$d2-d1=2\times dy \qquad (4)$$

式(3)と式(4)で使われているのはdxとdyだけで変数は含まれていません．つまり一度判定式を計算してしまえば，次の点はある値を足すだけでよいということになります.

実際のコーディング

これらの方法をCでコーディングしたものがリスト1です.

初期化の部分を除けばループの中身はわずか5行です.

（わかりやすくするために5行で書きましたが，リスト3なら1行だけ！）

疑似Cコードでループの内容を説明するとリスト2のようになります.

リスト3では，場合分けをちゃんとしてすべての傾きなどの条件の線分が描画できるように書き加えて，実際に動作するために必要なコードにしてあ

ワンラインNEWS
リコーはイーサネットに直接接続のできるレーザープリンタ「LP5100-UX」を発売した．LP5100-UXは解像度480dpsで最大A3版の用紙を毎分20枚まで出力できる.

ります.

Cが使える環境の人はリストを参考にプログラムを作って遊んでみてください.

やっぱりスピードは，アセンブラでガリガリ書いたBIOSとかハードウェアで実行するグラフィックプロセッサには勝てませんので，実用的な価値はないかも知れませんが，けっこうおもしろいものです.

目で見えるものというのはつい，凝ってしまいます……よね？

グラフィックのハードウェア

グラフィックのハードウェアとして，その専用のプロセッサが載っていてさまざまな図形の描画ができるようになっているものがありますが，そのハードウェアの中身もこのような方法をハードウェアロジックでやっているのでしょうね．いや，もしかしたら案外マイクロコードで行っていて，処理そのものは同じだったりして.

リスト1

```
void line(int x1, int y1, int x2, int y2){
    int dx, dy, d1, d2, d;

    dx = abs(x2 - x1);
    dy = abs(y2 - y1);
    d = 2 * dy - dx;
    d1 = 2 * dy;
    d2 = 2 * (dy - dx);
    plot(x1, y1);
    while(x1++ < x2){
        if(d < 0)
            d += d1;
        else
            y1++, d += d2;
        plot(x1, y1);
    }
}
```

リスト2

```
while(X座標が範囲内の時ループ，ついでにX座標を＋1){
        if(判定式が負なら)
                判定式に増分値1を加える;
        else
                Y座標を＋1して，判定式に増分値2を加える;
}
```

リスト3

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include <conio.h>

#define WIDTH   640
#define DEPTH   400
#define NCOLOR  8

#define PLANE0  0xa800
#define PLANE1  0xb000
#define PLANE2  0xb800

char far *plane0 = MK_FP(PLANE0, 0);
char far *plane1 = MK_FP(PLANE1, 0);
char far *plane2 = MK_FP(PLANE2, 0);

void plot(int x, int y, int color){
    int off, bit;

    if(x < 0 || WIDTH - 1 < x || y < 0 || DEPTH - 1 < y)
        return;
    off = (DEPTH - 1 - y) * (WIDTH >> 3) + (x >> 3);
    bit = 1 << (7 - (x & 7));
    plane0[off] = (color & 1)? (plane0[off] | bit): (plane0[off] & ~bit);
    plane1[off] = (color & 2)? (plane1[off] | bit): (plane1[off] & ~bit);
    plane2[off] = (color & 4)? (plane2[off] | bit): (plane2[off] & ~bit);
}

```

(注)

- PC-9801RA2で動作を確認しました.
- 8色モードです．16色モードは使っていません.
- Turbo-C用に書いてあります．ほかのCの場合には数か所修正が必要でしょう.
- パレットレジスタの設定をしていませんので，グラフィックを扱うほかのプログラムを動かした場合には注意が必要です.

```
31 void line(int x1, int y1, int x2, int y2, int c){
32     int dx, dy, d1, d2, d, dir;
33
34     dx = abs(x2 - x1);
35     dy = abs(y2 - y1);
36     if(dx >= dy){
37         d = (dy << 1) - dx;
38         d1 = dy << 1;
39         d2 = (dy - dx) << 1;
40         if(x1 < x2){
41             dir = (y1 < y2)? 1: -1;
42             for(plot(x1, y1, c); x1++ < x2; plot(x1, y1, c))
43                 d += (d < 0)? d1: (y1 += dir, d2);
44         }else{
45             dir = (y1 < y2)? -1: 1;
46             for(plot(x2, y2, c); x2++ < x1; plot(x2, y2, c))
47                 d += (d < 0)? d1: (y2 += dir, d2);
48         }
49     }else{
50         d = (dx << 1) - dy;
51         d1 = dx << 1;
52         d2 = (dx - dy) << 1;
53         if(y1 < y2){
54             dir = (x1 < x2)? 1: -1;
55             for(plot(x1, y1, c); y1++ < y2; plot(x1, y1, c))
56                 d += (d < 0)? d1: (x1 += dir, d2);
57         }else{
58             dir = (x1 < x2)? -1: 1;
59             for(plot(x2, y2, c); y2++ < y1; plot(x2, y2, c))
60                 d += (d < 0)? d1: (x2 += dir, d2);
61         }
62     }
63 }
64
65 void clearline(int n){
66     int i, off = n * (WIDTH >> 3);
67
68     for(i = 0; i < (WIDTH >> 3); i++)
69         plane0[off + i] = plane1[off + i] = plane2[off + i] = 0;
70 }
71
72 #define GBIOS_INTR  0x18    /* Graph-BIOS interrupt code */
73
74 #define GBIOS_START 0x40    /*  Start display */
75
76 #define GBIOS_STOP  0x41    /*  Stop display */
77
78 #define GBIOS_SETUP 0x42    /*  Setup */
79 #define GBIOS_RSV   0x0f    /*   (reserved = 0) */
80 #define GBIOS_BANK1 0x10    /*   0: Bank0, 1: Bank1 */
81 #define GBIOS_MONO  0x20    /*   0: Color, 1: Monochrome */
82 #define GBIOS_UPPER 0x40    /*   Display region - Upper */
83 #define GBIOS_LOWER 0x80    /*                    Lower */
84 #define GBIOS_ALL   0xc0    /*                    All */
85
86 void graphic(int on){
87     union REGS regs;
88     int i;
89
90     for(i = 0; i < DEPTH; i++)
91         clearline(i);
92
93     regs.h.ah = GBIOS_SETUP;
94     regs.h.ch = GBIOS_ALL;
95     (void)int86(GBIOS_INTR, &regs, &regs);
96
97     regs.h.ah = (on)? GBIOS_START: GBIOS_STOP;
98     (void)int86(GBIOS_INTR, &regs, &regs);
99 }
100
101 #define rand_x()    (rand() % WIDTH)
102 #define rand_y()    (rand() % DEPTH)
103 #define rand_c()    ((rand() % (NCOLOR - 1)) + 1)
104
105 int main(void){
106     int i;
107
108     graphic(1);
109     for(i = 0; !kbhit(); i = (i + 1) % DEPTH, clearline(i))
110         line(rand_x(), rand_y(), rand_x(), rand_y(), rand_c());
111     getch();
112     graphic(0);
113     return(0);
114 }
```

COLUMN　円の描画方法は……

線分の描画方法について説明しましたが，円も同様の方法で描画することが可能です．紙面の都合でアルゴリズムの説明は省略しますが，リスト3を円の描画に変えるための変更部分だけを示しておきます．

110行目のline () を呼びだしている部分を次のように変更してください．

```
circle (rand_x (), rand_y (), rand () % 100, rand_c ()) ;
```

次に右のリストにある関数cplot, circleを追加します．

どうですか？　計算時間を気にしないなら sin (), cos () を使ってしまいますが，このように単純なシフトと加減算だけで円が描画できるなんてちょっと不思議ですよね．

リスト■

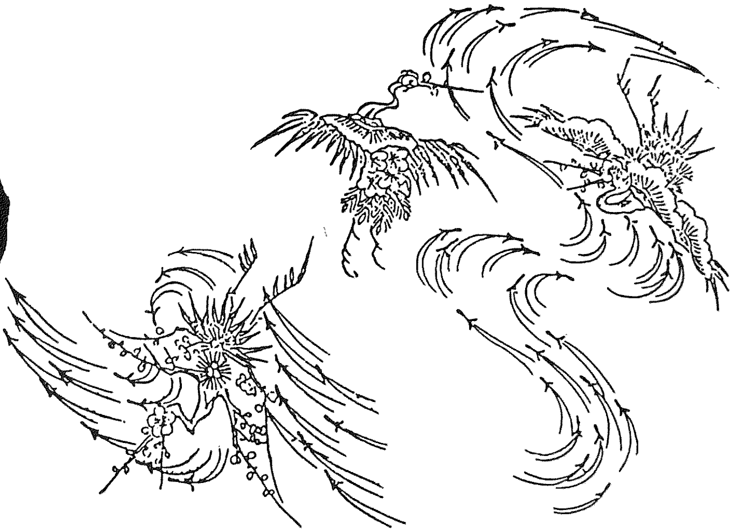
```
void cplot(int xc, int yc, int x, int y, int color){
    plot(xc + x, yc + y, color);
    plot(xc + x, yc - y, color);
    plot(xc - x, yc + y, color);
    plot(xc - x, yc - y, color);
    plot(xc + y, yc + x, color);
    plot(xc + y, yc - x, color);
    plot(xc - y, yc + x, color);
    plot(xc - y, yc - x, color);
}

void circle(int xc, int yc, int r, int color){
    int x, y, d;

    x = 0, y = r, d = 3 - (r << 1);
    while(x < y){
        cplot(xc, yc, x, y, color);
        if(d < 0)
            d += (x << 2) + 6;
        else
            d += ((x - y) << 2) + 10, --y;
        ++x;
    }
    if(x == y)
        cplot(xc, yc, x, y, color);
}
```


ワンラインNEWS
東レは書き込み済みのデータを消去しながら新たな情報を記録できる光ディスクを開発した．書き換え可能型の光磁気方式に比べ高速に動作する．

プログラム遊工房

谷口典生

弐乃巻
モアレ模様



今回は，グラフィック画面に幾何学模様を描き出すプログラムです（Macintosh用のSharewareであるMoireScreenSaverというスクリーンセーバーをまねして作ってみました）．単純なプログラムから美しい模様が生まれる様子を観察してください．

プログラムの中では，複数の変数をまとめて扱います．そのような場合は，構造体を利用することで変数をうまくまとめて，わかりやすく簡潔なプログラムを書くことができます．

リスト1

```
 1: /*
 2:  * moire.c       グラフィック画面にモアレ模様を描く
 3:  */
 4:
 5: #define TURBOC          0                /* Turbo C のときは 1 にする */
 6:
 7: #if TURBOC
 8: #include <graphics.h>
 9: #else
10: #include <graph.h>
11: #endif
12: #include <time.h>
13:
14: #define MAX_NUM         3                /* パターンを出す数 */
15: #define MAX_LEN         20               /* 図形を重ねる数 */
16: #define MAX_FIG         5                /* 図形の種類数 */
17: #define MAX_X           640              /* X 座標の最大値 */
18: #define MAX_Y           400              /* Y 座標の最大値 */
19: #define MAX_DX          30               /* X 座標の差分の最大値 */
20: #define MAX_DY          20               /* Y 座標の差分の最大値 */
21: #define MAX_COLOR       8                /* 色数 */
22:
23: struct figure {                  /* 一つの図形を定義する構造体 */
24:         int x1, y1;              /* 図形の座標その1 */
25:         int x2, y2;              /* 図形の座標その2 */
26:         int fig;                 /* 図形番号 */
27: };
28:
29: struct pattern {                 /* 一つのパターンを定義する構造体 */
30:         struct figure fig[MAX_LEN];     /* MAX_LEN 個の図形 */
31:         int color;               /* 色 */
32:         int dx1, dy1;            /* 座標その1を移動する差分 */
33:         int dx2, dy2;            /* 座標その2を移動する差分 */
34: } pat[MAX_NUM];                  /* MAX_NUM 個の構造体配列を宣言 */
35:
36: main()
37: {
38:         time_t t;
39:         int i, j;
40:
41:         time(&t);
42:         srand((unsigned int)t);          /* 乱数の初期化 */
43:
44:         init();                          /* 画面モードの初期化 */
45:
46:         for(i=0; i<MAX_NUM; i++) {
47:                 pat[i].fig[0].x1 = rand() % MAX_X;
48:                 pat[i].fig[0].x2 = rand() % MAX_X;
49:                 pat[i].fig[0].y1 = rand() % MAX_Y;
50:                 pat[i].fig[0].y2 = rand() % MAX_Y;
51:                 pat[i].fig[0].fig = rand() % MAX_FIG;
52:                 for(j=1;j<MAX_LEN;j++)
53:                         pat[i].fig[j] = pat[i].fig[0];
54:                 pat[i].color = 1;
55:                 pat[i].dx1 = rand() % MAX_DX - (MAX_DX/2);
56:                 pat[i].dx2 = rand() % MAX_DX - (MAX_DX/2);
57:                 pat[i].dy1 = rand() % MAX_DY - (MAX_DY/2);
58:                 pat[i].dy2 = rand() % MAX_DY - (MAX_DY/2);
59:         }
60:
61:         while(!kbhit()) {
62:                 for(i=0; i<MAX_NUM; i++)
63:                         drawpat(i);
64:         }
65:
66:         deinit();                        /* 画面モードの復帰 */
67: }
68:
69: drawpat(num)                     /* パターンを表示する */
70:         int num;                 /* パターン番号 */
71: {
72:         int i;
73:
74:         drawfig(&pat[num].fig[MAX_LEN-1], 0);   /* 最も古い図形を消去 */
75:         for(i=MAX_LEN-1; i>0; i--)
76:                 pat[num].fig[i] = pat[num].fig[i-1];
77:         pat[num].color = pat[num].color % (MAX_COLOR-1) + 1;
78:
79:         drawpatsub(&pat[num].fig[0].x1, &pat[num].dx1, MAX_X-1);
80:         drawpatsub(&pat[num].fig[0].x2, &pat[num].dx2, MAX_X-1);
81:         drawpatsub(&pat[num].fig[0].y1, &pat[num].dy1, MAX_Y-1);
82:         drawpatsub(&pat[num].fig[0].y2, &pat[num].dy2, MAX_Y-1);
83:
84:         if((rand() % 100)==0) {                  /* 時々図形と速度を変える */
85:                 pat[num].dx1 = rand() % MAX_DX - (MAX_DX/2);
86:                 pat[num].dx2 = rand() % MAX_DX - (MAX_DX/2);
87:                 pat[num].dy1 = rand() % MAX_DY - (MAX_DY/2);
88:                 pat[num].dy2 = rand() % MAX_DY - (MAX_DY/2);
89:                 pat[num].fig[0].fig = rand() % MAX_FIG;
```

プログラムの使い方

このプログラムはPC-9801シリーズ用で，Turbo C，Quick CおよびMS-Cに対応しています．Quick CまたはMS-Cでコンパイルする場合は，リスト1とリスト2を打ち込んでください．Turbo Cを使う場合は，リスト1とリスト3を打ち込み，リスト1の5行目の「#define TURBOC 0」を「#define TURBOC 1」にしてください（Turbo CとMS-C（Quick C）ではグラフィック関数が大きく異なるので，drawfig ()，init ()，deinit () の3つの関数が別リストになっています）．

今回のプログラムではグラフィック関数を使っているので，MS-CやQuick Cでコンパイルするときにはグラフィックライブラリを指定します．MS-Cなら「CL moire.c /link graphics.lib」などとします．Turbo Cでは実行時に.BGIファイルが必要となるので注意してください（詳しくはプログラムの説明のinit () の部分を見てください）．

MS-DOSのプロンプトから

MOIRE

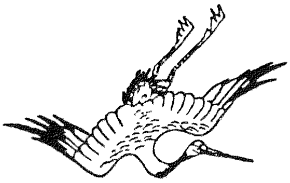
として起動すれば，グラフィック画面に幾何学的な連続パターン（モアレ模様）が描かれていくはずです．なにかキーを押せば終了します．

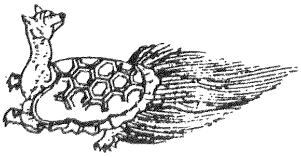
直線や四角形，楕円などの図形が重なったパターンが3つ画面を動いていくようすがわかると思います．定数を変更して（定数についてはプログラムの説明

の項で詳しく説明します)，模様の変化を楽しんでください．重なる数を減らして速度を大きくするとスピード感が出ますし，パターンの数を減らせばもっとスピードが上がります．

定数を変更するたびにいちいちコンパイルしなおすのがめんどうだと思ったら，それらの定数を変数にして，起動するときにパラメータで指定できるようにプログラムを書き換えてみましょう．それほどむずかしくはないはずです．ぜひ挑戦してみてください．

さらに余力がある人は，このプログラムを常駐させて，「コマンドラインで一定時間キー入力がなければスクリーンセーバーとして起動する」ようにしてみましょう．常駐させるためには工夫が必要なので，少しむずかしいかもしれません．常駐型のプログラムについてはいずれこのコーナーでもとりあげるつもりなので，楽しみ（?）にしていてください．

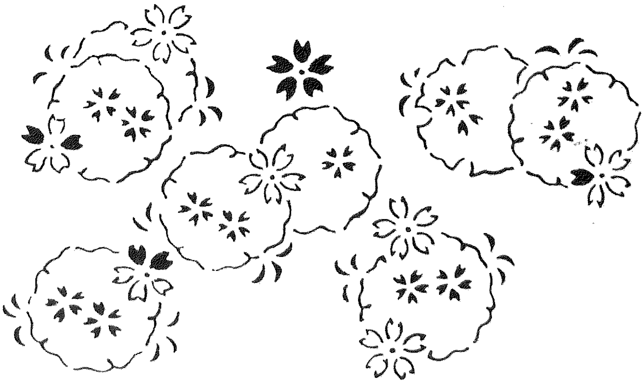




```
                }
        drawfig(&pat[num].fig[0], pat[num].color);      /* 新しく図形を描く */
}

drawpatsub(val, dv, max)        /* 座標が画面の範囲内に収まるようにする */
        int *val, *dv, max;
{
        *val = *val + *dv;
        if(*val<0) {
                *val = 0;
                *dv = -*dv;
        } else if(*val>max) {
                *val = max;
                *dv = -*dv;
        }
}
```

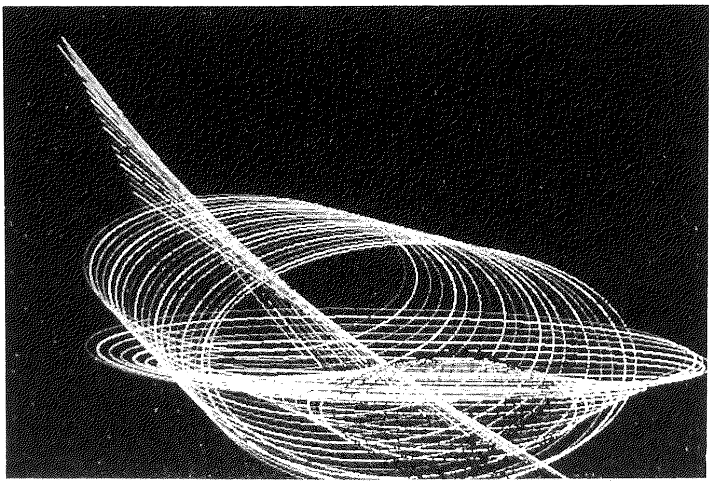
リスト2

```
/*
 * MS-C, Quick C 用の drawfig(), init(), deinit()
 */

drawfig(fig, col)               /* 図形を表示する */
        struct figure *fig;
        int col;
{
        _setcolor(col);
        switch (fig->fig) {
                case 0 :        /* 直線 */
                        _moveto(fig->x1, fig->y1);
                        _lineto(fig->x2, fig->y2);
                        break;
                case 1 :        /* 長方形 */
                        _rectangle(_GBORDER,fig->x1,fig->y1,fig->x2,fig->y2);
                        break;
                case 2 :        /* V 字型 */
                        _moveto(fig->x1, fig->y1);
                        _lineto((fig->x1+fig->x2)/2, fig->y2);
                        _lineto(fig->x2, fig->y1);
                        break;
                case 3 :        /* 砂時計型 */
                        _moveto(fig->x1, fig->y1);
                        _lineto(fig->x2, fig->y2);
                        _lineto(fig->x1, fig->y2);
                        _lineto(fig->x2, fig->y1);
                        _lineto(fig->x1, fig->y1);
                        break;
                case 4 :        /* 楕円 */
                        _ellipse(_GBORDER, fig->x1, fig->y1, fig->x2, fig->y2);
                        break;
        }
}

init()          /* 画面モードの設定 */
{
        _setvideomode(_98RESCOLOR);     /* グラフィック8色モード */
}

deinit()        /* 画面モードの復帰 */
{
        _setvideomode(_DEFAULTMODE);    /* デフォルト(テキスト80行)モード */
}
```



リスト 3

```
/*
 * Turbo C 用の drawfig(), init(), deinit()
 */

drawfig(fig, col)                   /* 図形を表示する */
        struct figure *fig;
        int col;
{
        setcolor(col);
        switch (fig->fig) {
                case 0 :        /* 直線 */
                        line(fig->x1, fig->y1, fig->x2, fig->y2);
                        break;
                case 1 :        /* 長方形 */
                        rectangle(fig->x1, fig->y1, fig->x2, fig->y2);
                        break;
                case 2 :        /* V 字型 */
                        moveto(fig->x1, fig->y1);
                        lineto((fig->x1+fig->x2)/2, fig->y2);
                        lineto(fig->x2, fig->y1);
                        break;
                case 3 :        /* 砂時計型 */
                        moveto(fig->x1, fig->y1);
                        lineto(fig->x2, fig->y2);
                        lineto(fig->x1, fig->y2);
                        lineto(fig->x2, fig->y1);
                        lineto(fig->x1, fig->y1);
                        break;
                case 4 :        /* 楕円 */
                        ellipse((fig->x1+fig->x2)/2,(fig->y1+fig->y2)/2,0,360,
                                abs(fig->x1-fig->x2)/2,abs(fig->y1-fig->y2)/2);
                        break;
        }
}
init()          /* 画面モードの設定 */
{
        int graphdriver, graphmode;

        graphdriver = DETECT;           /* 自動検出 */
        graphmode = PC98C8;             /* 8色モード */
        initgraph(&graphdriver, &graphmode, "");
}

deinit()        /* 画面モードの復帰 */
{
        restorecrtmode();               /* 画面モードを元に戻す */
}
```



プログラムの説明

14-21行

各種定数を定義します.

14行

MAX_NUM はパターンの数です.多くしすぎるとわけがわからなくなるので, 1から3程度が適当だと思います.

15行

MAX_LENの数だけ図形が重なってパターンを構成します.この数を大きくするとそれだけ尾を長くひくように見えるわけです.

16行

MAX_FIGは使う図形の種類です.関数drawfig()を見ればわかるように,図形番号0から4までがそれぞれ直線,長方形,V字型,砂時計型,楕円に対応しています.楕円は描画に時間がかかるので,グラフィック関数が遅いTurbo CではMAX_FIGの値を4にして,楕円は使わないほうがよいかもしれません.

17-18行

XY座標の最大値(座標は0から始まるので,正確には最大値+1)です.PC-9801シリーズのノーマルモードではグラフィック画面のサイズは640×400なので,それぞれ640と400になっています.

19-20行

XY座標それぞれを動かす速度の最大値です.この値を大きくすれば動きは速くなりますが,模様は粗くなります.小

さくすると動きは遅く，連続度の高い模様になります．

21行

色数です．実際には1からMAX_COLOR−1までのカラーコードが使われるので，MAX_COLOR−1の色数で表示されます．16色モードを使う場合にはMAX_COLORを16にして，init ()関数で「_setvideomode (_98RES16COLOR)；」(MS-C，Quick C)または「graphmode ＝ PC98C16;」(Turbo C) とします．このときはパレットを設定してうまく色が連続して変化するようにすると美しい模様が見られると思います．

❖main ()

41-42行

time関数から得た値を利用して乱数を初期化します．これをしないと，プログラムを起動するたびにいつも同じパターンで図形が描かれてしまいます．

44行

画面モード（グラフィックモード）を設定します．Turbo Cの場合は，BGIを初期化します．

46-59行

各パターンの初期値を設定します．図形の座標と種類，速度（XY座標の差分）をランダムに決めます．各図形データ(pat[].fig[0]から pat[].fig[MAX LEN-1]）には同じ値を入れておきます．

61-64行

このプログラムのメインループです．なにかキーが押されるまで，それぞれのパターンを描き続けます．

66行

画面モードを元に戻します．Turbo Cの場合はBGIを終了します．

❖drawpat ()

74行

pat[].fig[0]からpat[].fig[MAX LEN-1] には新しく描いたものから順に図形のデータが入っています．もっとも古い図形をカラーコード0（黒）で上書きして，消去します．

75-76行

図形データを順に後ろへずらしていきます．画面から消去したもっとも古い図形データがなくなって，一時的に図形番号0と図形番号1のデータが同じになることに注意してください．

77行

図形を描くカラーコードを更新します．4096色中16色モードを使って，うまくパターンがグラデーションを描くようにパレットを設定すれば，より美しい模様になります．

79-82行

図形番号0の図形の座標を更新します．座標が画面の範囲からはみ出さないようにサブルーチンを使って処理しています．

84-90行

いつも同じ図形ではつまらないので，ときどき（ここでは100回に1回の割合でランダムに）図形の種類と速度を変えています．

91行

データが更新された図形番号0の図形を画面に表示します．

❖drawpatsub ()

97-104行

図形のXY座標に差分を加え，新しい座標が画面の範囲を越えた場合には画面の端の座標に戻し，差分の符合を逆にして，図形の進む方向を反転させます．これによって，画面の端にぶつかった図形は壁に反射するような動きをすることになります．

❖drawfig ()

図形番号（0～4）に対応した図形を実際にグラフィック画面に描きます．座標1（x1，y1）と座標2（x2，y2）に囲まれた範囲に各図形を描いています．前述したように，楕円は描画に時間がかかるため，楕円のパターンが2つ以上になったりすると急にスピードが遅くなります．楕円のかわりにほかの直線で構成される図形を描くようにするのもおもしろいと思います．case分を増やし，MAX_FIGの値を増やせばもっと多くの図形を使うこともできます．X字型，ダイヤ型，星型などいろいろ工夫してみるとよいでしょう．

❖init ()

画面モードの設定を行います．Turbo CではBGI(Borland Graphics Interface) の初期化を行っていますが，このままコンパイルして作った.EXEファイルはカレントディレクトリに.BGIファイル(PC98.BGI, PC98EGC.BGI，PC98GRCG.BGI) がないと動きません．ほかのディレクトリから起動したい場合は，「initgraph (&graphdriver，&graphmode，"")；」の最後の引数「""」に．BGIファイルのあるパス名を指定してください．たとえば，通常のインストールで「A:¥TC」にTurbo Cがインストールされているなら，「initgraph(&graphdriver，&graphmode，"A:¥¥TC")；」(文字列の中で¥は¥¥と表記することに注意) とすればどこからでも起動することができます．

❖deinit ()

Turbo CではBGIの終了，MS-C (Quick C) では画面モードの復帰を行います．

ワンポイントテクニック

構造体を使ってプログラムをスッキリと

今回のプログラムで，パターンを構成している変数を考えてみましょう．

まず各図形は２つの点によって決まるので，１つの図形について図形番号とX座標，Y座標が２つずつで合計５つの変数が必要です．１つのパターンはMAX_LEN個の連続した図形によって描かれるので，MAX_LEN個分のデータが必要です．ほかに色と速度（ふたつのXY座標を移動させる差分値）のデータもいります．そして，そのパターンのデータがMAX_NUM個あればよいわけです．

普通の配列だけを使って変数を宣言してみると，図１のようになります．リスト１の構造体を使った宣言（23行から34行）と比べてみてください．構造体を使ったことがない人にはかえって複雑に見えるかもしれませんが，慣れればリスト１のように構造体を使ったほうがデータ構造もわかりやすいはずです．図１のように宣言した変数と，リストでの構造体を使った変数とを対応させると，図２のようになります．構造体名とメンバ名の間は「.」で区切ります．

構造体を使う最大の利点は，まとまったデータを１つの構造体名で扱うことができることです．リスト１の52，53行を見てください．図１のように宣言した変数で同じ部分のプログラムを作ると図３のようになり，かなり冗長なプログラムになってしまいます．リスト１の75，76行の部分についてもまったく同様です．

関数にこれらのデータを渡すときはどうでしょうか．drawfig () という関数は引数に図形のデータと色を取りますが，変数を図１のように宣言した場合は６つもの引数をとる関数になってしまいます．リストでは図形のデータ５つが構造体figureとしてまとまっているので，引数は２つですんでいます．

構造体を関数に渡すときはアドレスを渡し，関数側では構造体名とメンバ名の区切りに「->」を使います．つまり，関数の呼び出し側では「func (&pat [n]. fig [1])」と渡し，受け取る側は図４のように処理します．つまり，「fig-> x1」は「(*fig). x1」と同じなのです．

プログラムを作るときには，データ構造をうまく決めることが重要です．データ構造さえきちんと決めてしまえば，あとは楽にプログラムを書くことができます．逆にデータ構造がいいかげんなまま，いきあたりばったりでプログラムを書きはじめてしまうと，あとで手直しするのが困難になったりするものです．

今回のプログラムのように複数の変数をまとめて扱うような場合には，構造体を利用すれば変数をすっきりとまとめてわかりやすくすることができるし，プログラムも簡潔に書くことができます．

図１　配列だけを使った変数宣言

```
int pat_x1[MAX_NUM][MAX_LEN];   /* 図形の X 座標その 1 */
int pat_y1[MAX_NUM][MAX_LEN];   /* 図形の Y 座標その 1 */
int pat_x2[MAX_NUM][MAX_LEN];   /* 図形の X 座標その 2 */
int pat_y2[MAX_NUM][MAX_LEN];   /* 図形の Y 座標その 2 */
int pat_fig[MAX_NUM][MAX_LEN];  /* 図形番号 */
int pat_col[MAX_NUM];           /* 色 */
int pat_dx1[MAX_NUM];           /* X 座標その 1 を移動する差分 *
int pat_dy1[MAX_NUM];           /* Y 座標その 1 を移動する差分 *
int pat_dx2[MAX_NUM];           /* X 座標その 2 を移動する差分 *
int pat_dy2[MAX_NUM];           /* Y 座標その 2 を移動する差分 *
```

図２　配列だけを使った変数と構造体を使った変数の比較

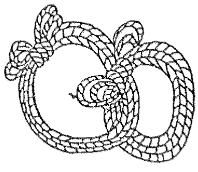
```
pat_x1[ n ][ 1 ]    =    pat[ n ].fig[ 1 ].x1
pat_x2[ n ][ 1 ]    =    pat[ n ].fig[ 1 ].x2
pat_col[ n ]        =    pat[ n ].col
pat_dx1[ n ]        =    pat[ n ].dx1
```

図３　配列だけを使ったプログラム

```
for(j=1;j<MAX_LEN;j++) {
        pat_x1[ j ] = pat_x1[ 0 ];
        pat_y1[ j ] = pat_y1[ 0 ];
        pat_x2[ j ] = pat_x2[ 0 ];
        pat_y2[ j ] = pat_y2[ 0 ];
        pat_fig[ j ] = pat_fig[ 0 ];
}
```

図４　構造体の受け取り

```
func(fig)
        struct figure *fig;
{
        fig->x1 = ....
}
```



情報処理技術者試験の動向

赤 松 高 文
東京コンピュータサービス株式会社

急増する受験者 相変わらず低い合格率

先月号では，情報処理技術者試験の沿革・区分，そして合格した場合のメリットについて述べられていました．それでは次に，実際に受験する際気にかかる，受験者数・合格率等について調べてみましょう．

表1，図1に示すとおり，年間50万人に近い応募者があります．これは，国家試験としては，自動車の運転免許につぎ，最大規模のものです．さて，次には肝心の合格率です．第1種・第2種では10%台，特種・システム監査で10%以下，平成1年のオンラインに至っては，やっと2%の合格率しかありません．さらに注意してもらいたいのは，この合格率が受験者数に対するものであることです．出願者数に対する合格率となるとさらに低下します．せっかく応募して，受験料3,600円也を支出しても，実際に受験する人は，その6割しかいないのです．情報処理技術者試験に合格した場合のメリットがこれだけ強調され，毎回多数の出願者がありながら，低い受験率・さらに低い合格率というのが，現実の姿だといえましょう．

図1　情報処理技術者試験の推移表

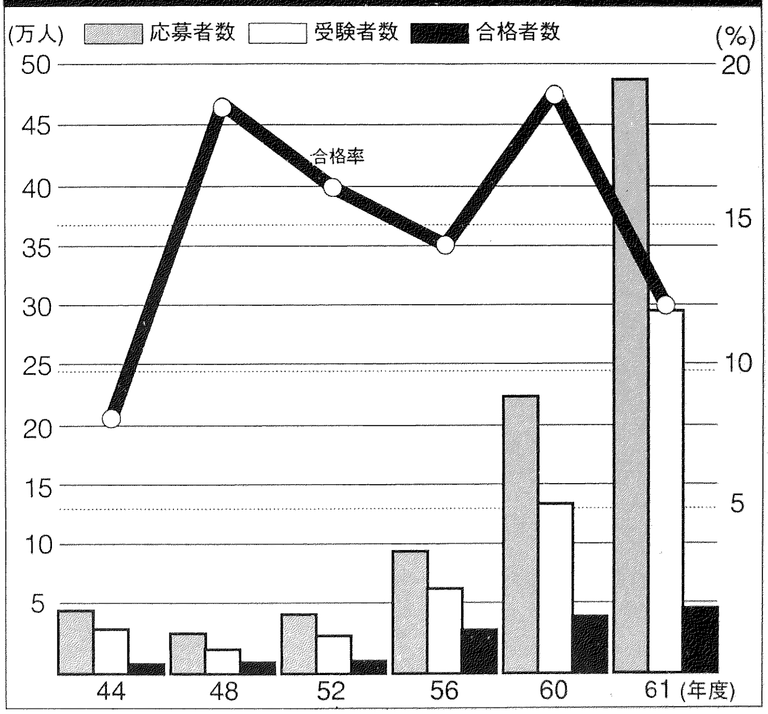


表1　種目別受験者状況(平成1年度)

	システム監査	特種	オンライン	1種	2種	合計
応募者数	9,444	28,106	36,776	76,969	331,757	483,052
受験者数	5,213	14,947	20,274	45,339	204,793	290,566
受験率(%)	55.2	53.2	55.1	58.9	61.7	60.2
合格者数	358	1,245	409	4,766	27,874	34,652
合格率(%)	6.9	8.3	2.0	10.5	13.6	11.9

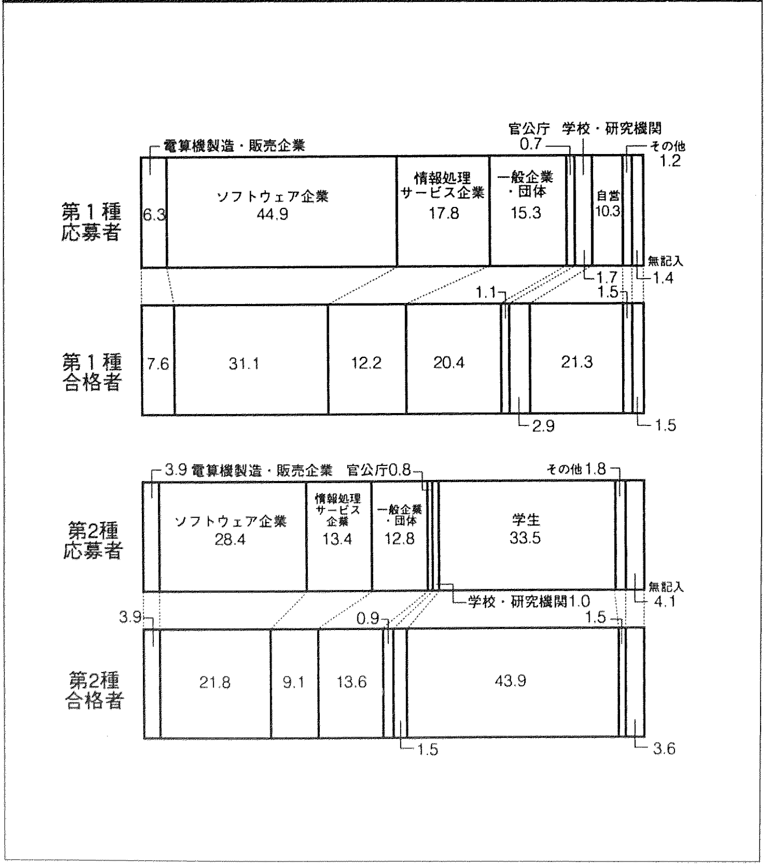
なぜ，合格率は低いのだろうか

情報処理技術者試験は，資格試験ではないので，合格しないと仕事に従事できないわけではありません．それでも，大多数の企業では，合格者に対して，合格一時金を支給したり，月々の給与に資格手当をつけたりしています．あるいは，昇級・昇進の規準や，考課の重要な目安にしている企業は少なくないようです．このように，メリットがあるにもかかわらず，なぜ，受験率・合格率は低いのでしょうか．

図2をご覧ください．応募者の中に占める学生の割合が意外と高いことに気がつきます．さらに合格者となると，その割合がぐっと増加します．この理由として，実際にソフトウェア開発に携わる者が，不合格の言い訳にする「試験勉強なんかしている暇がない」というのがあげられるでしょう．

私自身の場合も，1か月の残業時間が200時間を越えた，などということがあります．過労死しないのが不思議なぐらいです．こんなことは始終ではないにしても，ソフトウェア開発の業界では，技術者の残業は月平均40～50時間といったところではないでしょうか．そうすると，自宅に帰っても，あるいは通勤の車中でも（座れるとして），参考書を開くよりは睡眠時間を確保する方が先決，ということになりがちです．そして結局は，会社の上司に命令され，出願はしてみたものの，勉強不足でどうせ合格する見込みはない．そうして今回も受験しなかった，となるわけです．また，仕事が忙しければ，試験当日は休日出勤となることもあるでしょう．逆に，たまたま仕事が落ちついた時期と試験日が一致しても，どうせ一夜漬けでは無理だからと，その週は休暇を取って，F1グランプリを見にいってしまうということになるのです．いくら企業の側がはっぱをかけても，現場の技術者には，試験勉強に取り組むだけの気力が残っていないのかもしれません．

図2　勤務先別構成比（平成2年度春期）



企業は合格率を上げる努力をしているか

受験者の側だけでなく，企業の立場からも考えてみましょう．多くの企業が，情報処理技術者試験の合格者に対して，それなりの評価を与え，なんらかの特典はつけています．その理由のひとつは，技術者を評価する基準に他に適当なものがないからでしょう．しかし，それだけではありません．企業の側にも，情報処理技術者試験合格者がいることは，大きなメリットになるのです．

ソフトウェア開発業界では，1986年に派遣法が施行されるまで，ソフトウェア技術者を派遣することが主流でした．技術者を派遣するときに，その技術者が情報処理技術者試験に合格している場合，より高い金額で契約することができます．また，最近では，通産省の認定するシステムインテグレータになることが，ソフトハウスのトレンドであると考えている企業もあります．システムインテグレータに認定されるためにも，ソフトウェア開発会社としての実力を示すためにも，社内により多くの情報処理技術者試験合格者のいることが条件となってきます．そこで，企業としては，さまざまな恩恵

ワンラインNEWS
富士通は64KビットのデュアルポートSRAMを開発．アクセス速度は45ナノ秒でデュアルポートRAMとしては最大容量．4月から月産20万個を予定している．

を与えて，社員の合格率を高めようとする訳です．ところが，先にも触れたように，社員には気力が失せて，笛吹けども踊らずという状況にあるのです．

ただ特典を与えて，自主的に勉強するのを待っていたのでは，埒があかない．そこで，次に教材を与えて勉強をさせようということになります．

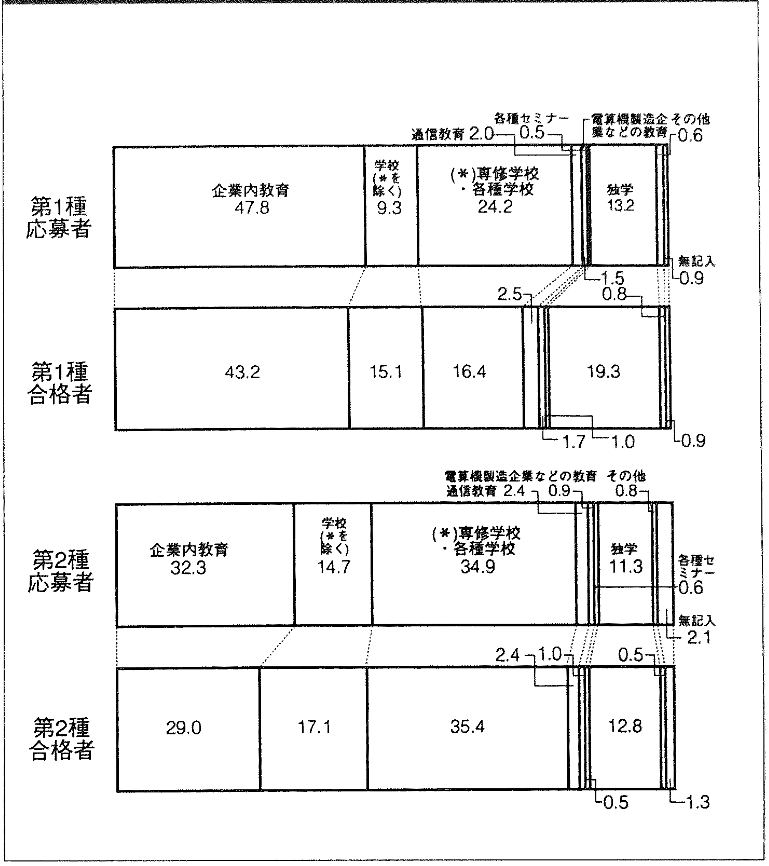
参考書・講座など受験産業花盛り

大きな書店のコンピュータ・コーナーの一角には，必ず情報処理技術者試験の棚や台があるものです．そこには，受験参考書・受験雑誌が幾種類も並んでいます．また，短波放送では受験講座も開かれていますし，各種専門学校も受験講座や通信教育を開設しています．この中で，通信教育講座について少し触れてみます．

一般的な情報処理技術者試験対策通信講座は，5～6か月の期間で，3万円程度の価格です．これには，受験の手引き・ソフトウェアの知識・ハードウェアの知識・関連知識・言語の学習（COBOL・FORTRANまたはCASL）といったテキストと，テキスト毎の添削テスト，それに模擬テストが1～2回含まれます．さらに，価格は上がりますが，講義カセット・テープをつける講座もあります．また，別に講義のビデオ・テープも販売されています．

図3に見られる独学者は，参考書・受験雑誌・通信講座のいずれか，または全部で勉強したものと思われます．さらに，企業内教育のうち，かなりの数は，市販の通信講座に頼っているのが実情ではないでしょうか．

図3　研修先別（情報処理関係）構成比（平成2年度春期）



効果の上がらない企業の教育投資

もう一度，図3を見てください．企業内教育では，応募者に対する割合は高いのに，合格者に対する割合では低くなっています．その原因としては，「勉強する暇がない」24時間戦うソフトハウスマンの実態があげられるでしょう．しかし，それだけではないと思われます．ソフトウェア業界に限らず，目先の利益にとらわれて，教育に投資したがらない企業の体質が見えてきます．教育の成果はすぐには現れませんし，評価の方法も難しいかもしれません．そこで教育担当者を選ぶときに，つい「あいつは仕事ができない，売上高も低いから，教育でもやらせておけ」となってしまうこともあります．これでは教育の成果は上がらず，合格率も上昇しようがありません．また，まともな人材（？）を教育担当者にしても，経験がないのでひとりよがりになってしまったり，自社の実情にあわない業者の教育に頼りがちです．そこで最近浮上してきたのが，情報処理技術のインストラクター（指導者）制度です．

インストラクター認定制度91年をめどに新設へ

何度も触れているとおり，ソフトウェア技術者はたいへん忙しい人種です．その背景には，深刻なソウトウェア技術者の人材難があります．通産省によれば，「西暦2000年までに97万人のソフトウェア技術者が不足する」と

いうことになります．この事態を打開する方策として，情報処理技術者試験を通じての，技術力の向上と社会への認知が行われているわけです．しかし，一方では，指導者の不足も大きな問題となっているのです．

いまだに一部の企業では，入社案内に「未経験者の方でも新入社員教育を行いますので安心です」と載せていながら，その実態は，マニュアルを一冊渡して「これ読んどいて」で終わってしまうところもあるようです．

そこで通産省では，91年をめどに，インストラクターの認定制度を新設することになりました．そのため，従来では受験講座ばかりだった受験産業の中に，「情報処理技術者のためのインストラクター養成講座」が加わってきています．

情報処理技術者試験は早目に合格しよう

図4を見ましょう．第1種も第2種も，合格者の大多数が20代です．第2種では，19歳のところにピークがありますが，これは，専門学校の1年目で，情報処理技術者試験の受験コースを受講するためでしょう．私の会社でも，入社前に情報処理技術者試験2種に合格している者が年々増加しています．21歳というのは，専門学校卒で入社1年目，24歳というのは，大学卒で入社後1～2年目に合格するケースが多いということを示しています．

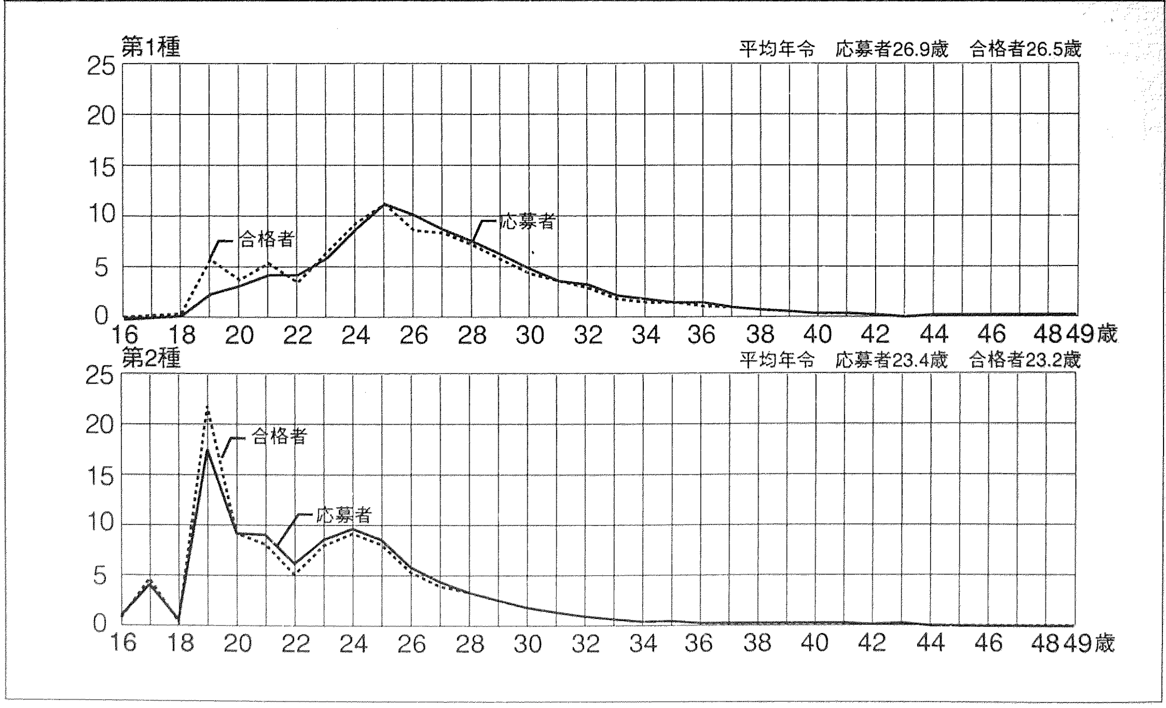
第1種になると，第2種に合格してから受験する人が多く，さすがに年齢は上がります．それでも，合格者は25歳のほかに19歳にもピークがあり，20歳代後半以後は応募者とともに減少していきます．

この原因のひとつ目は，ソフトウェア技術者は，まあ1年目は勉強させてもらえても，2年目以降はどんどん仕事が増加する．そのうえ，経験年数によって，その技術者の能力に関係なく，プロジェクト・マネジメントも任され，ますます仕事が多くなる．さらに，ソフトウェア開発では，マメジメント技法がなっていないと，悪条件が重なり，どんどん情報処理技術者試験から遠ざかる——ということがあげられます．

原因のふたつ目としては，「そのうち受験しよう」と思っているうちに，どんどん後輩の技術者が合格してしまい，「いまさら1種や2種でもあるまい」と，実力以上のものを受験して玉砕することがあげられるでしょう．

もう一度，図4を見直して，早期合格を目指して，今日から勉強いたしましょう．

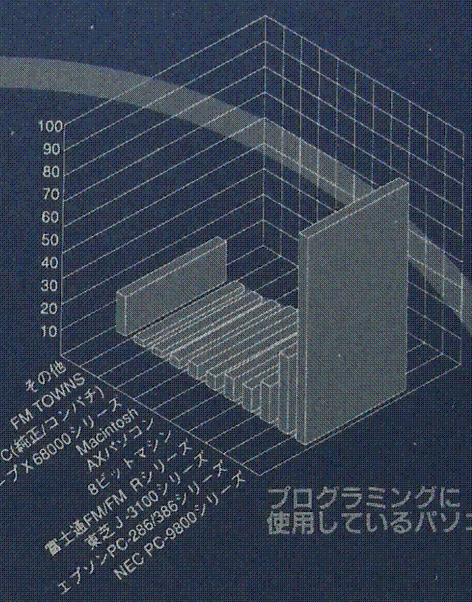
図4　年齢別構成比（平成2年度春期）



ワンラインNEWS
セガエンタープライズが今春発売を予定しているパソコン「テラ」にはZ80，68000，8086とCPUが3つも搭載されている．

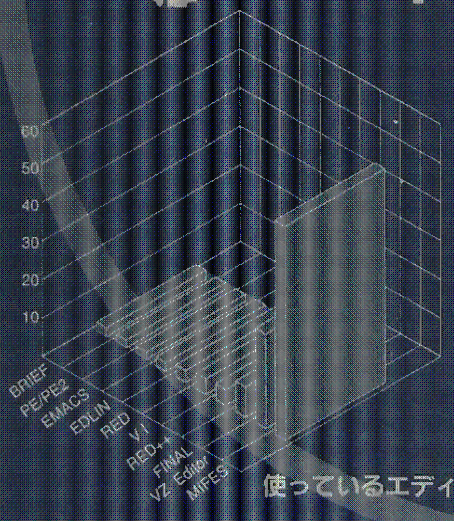
データ・ページ'91

このデータは，本誌創刊にあたって，プロ・アマチュアのプログラマを対象に実施したアンケート結果を集計したものです（有効回答者数：306名）．



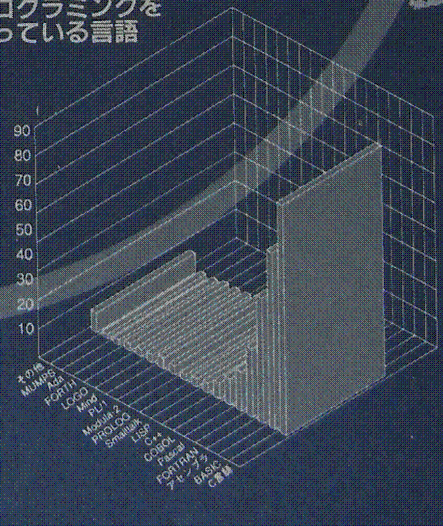
プログラミングに使用しているパソコン

information・page 2



使っているエディタ

プログラミングを行っている言語



Book Review

■中村正三郎

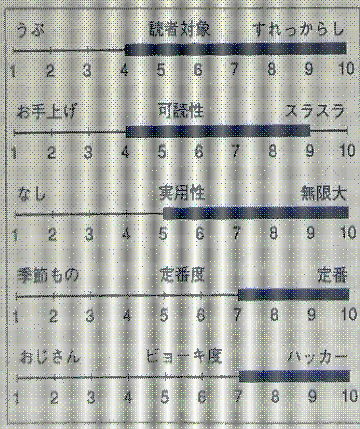
プログラマならぜひ座右にほしい コンパイラ教科書の定番

■コンパイラⅠ，Ⅱ

原理・技法・ツール

A.V.エイホ，R.セシィ，J.D.ウルマン共著
原田賢一 訳
サイエンス社　全916ページ
Ⅰ，Ⅱ巻各5768円

■評価チャート



待たれて久しかった「Compilers Principles, Techniques, and Tools」の待望の日本語訳である．コンパイラ技術全域に渡って深く解説してあり，コンパイラの教科書の定番といってよい内容を備えている．プログラマならぜひ座右にほしい本である．原書は表紙の絵にちなんで，ドラゴンブックと呼ばれる．訳者，編集者がそれを尊重してくださったのだろう，うれしいことに本書でもドラゴンは健在だ．翻訳も読みやすい．私は本書を「無条件にお薦めする」と書いてしまえば，今月はもう書くことがない．さっそく字数稼ぎをせねばならない．

最近の洋書はやたらと分厚く500ページを超えるものは珍しくない．訳すとページ数が増えるのか，日本語版では分冊になることがよくある．本書も例外ではなく約800ページの原書を2冊に分けている．私はあまり分冊が好きではない．本書はⅠ，Ⅱが同時に出たから偉い(関係者の努力の賜物)．だが，中にはⅠだけ出てⅡがなかなか出ないことがある．これがいやなのだ．Ⅰだけ出して様子をみて，Ⅱの印刷部数を調整するのかもしれない．製本上の困難や読みやすさの問題があるのは承知しているつもりだが，できれば1冊にしてほしいものだ．ちなみに，本書の原書のハードカバーは，ひと回り版が大きいとはいえ，分冊になった本書の1冊より少し厚い程度である．もっと薄くていい紙を使うとか，編集者の腕の見せどころはあると思うのは，素人考えだろうか．

先にぜひ座右にと書いたが，値段はとても高い．1冊6000円見当，2冊で1万2000円で少しお釣がくる．個人で無理なら会社に買わせること．計算機関係の学生ならデートを我慢し，バイトをしてでも買っておいた方がいい．なお，原書のハードカバーは1万円程度，ペーパーバックは3500円程度で買える．英語が苦にならないのなら，後者がねらい目だ．アメリカではこのように廉価版を出すことがよくあるが，なぜか日本ではやらない．どなたか理由を教えてください．

本書の読者対象は，多少なりともプログラミングの経験がある人だ．アルゴリズムやデータ構造の知識は必要になる．Pascalの疑似コードでアルゴリズムを，Cでサンプルコードを示しているので，これらの言語の知識は少しだけいる(精通している必要はない)．

本書の内容に簡単に触れる．全部で12の章と付録があり，以下のとおり．

1 概説／2 簡単な1パスコンパイラ／3 字句解析／4 構文解析／5 構文主導翻訳／6 型検査／7 実行時環境／8 中間コード生成／9 コード生成／10 コード最適化／11 コンパイラの開発／12 コンパイラの実現例／付録 プログラム作成演習．

1，2章は概要で，ここで述べた各項目について3から10章で深く掘り下げる．各章は，初級から徐々に中級から上級の話題となる．式のコンパイル程度なら1，2章を読めばよい．これと各章のはじめを拾い読みすれば，初級から中級程度のコースとなる．11章はコンパイラ開発時の一般的アドバイス．12章は実在する各種コンパイラの説

ワンラインNEWS
Windowsの機能強化について米Microsoft社は1991年前半にはROM化が可能なWindows3.1をリリース，1991年後半には手書き文字認識機能の拡張をおこなうと発表

明で，これは読物としてもおもしろい．

コンパイラの本では，字句解析，構文解析などコンパイル前半部に重点をおき，コード生成，最適化など後半部にページを割いていないものがよくある．その点本書は，8，9，10章でそれらを詳しく解説している．型検査について章を別に設けているのも目を引く．

コンパイラに使われている技術は，その他のプログラミングにも役立つ．式の評価(例 スプレッドシートやデータベースの計算項目)，各種コマンド解析（例：コマンドインタプリタ，エディタやフォーマッタのコマンド），機能拡張のためのマクロ言語，文字列検索，表探索などに使うことができる．学んでおいて損はない．再び言う．私は本書を「無条件にお薦めする」．

ウィンドウ関連書籍の訳本が続々登場

古田島義和（神田・書泉グランデ）

GUIの革新的な開発がすすみ，マルチウィンドウという言葉が身近になってきました．

89年末ころからGUIが脚光を浴びはじめ，ワークステーションの世界ではOSF/MotifとUI/OpenLookが代表的です．これに関する図書として，「OSF/Motif全5巻シリーズ」（Prentice Hall，以下P. H）と「OpenLook全2巻シリーズ」（Addison Wesley，以下A. W）がロングセラーになっています．このうちOpenLookは，A. W社で翻訳出版をしました．一方P. H社のOSF/Motifは，バージョンの関係で訳本がのびているようです．ちなみにA. W社でも，MinkleyとHolt共著で「programming with OSF/Motif」の出版が近く予定されています．

パソコンの世界でも，90年夏ころからGUIが注目を集めだし，ほとんどの米国雑誌がWindows 3.0の特集記事を載せました．日本電気などが今年早々にVer. 3.0の日本語版を出荷するという話もあります．

Windows 3.0の図書はまだ洋書ばかりで，「Windows programming」「Using Window 3.0」（2冊ともMc Graw Hill）と「Mastering Windows 3.0」（Sybex）などが売れています．このほかにMicrosoft Press社の全3巻シリーズ「Guide to Programming」「Progarmmer's Reference」「Programming Tools」と「Companion」「Running Windows 3/E」が入荷しベストセラーになっています．

X Windowについては，書き下ろし・訳本が短期間のうちにいろいろ出版されました．日刊工業新聞社から「実用X Window」「X Window Ver. 11プログラミング」などが出版され，「X Window OSF/Motifプログラミング」（日刊工業新聞社）と「X Viewプログラミング入門」（スペック／工学図書）がベスト10の2位と3位を占めました．日刊工業新聞社は「入門X Window」を出版し，書き下ろしシリーズを締めくくりました．

訳本では，「X Windowハンドブック」（アスキー），「X Toolkitプログラミング」（トッパン）が好評です．トッパンの第2弾として，「X Window System programming and Application with Xt/Motif Edition」（P. H社）の訳本が「X Toolkitプログラミング－OSF/Motif版」として近日出版されます．おおいに期待する1冊です．

洋書のほうでは，一段落ちついた感じですがベスト5位に「Advanced X Window Application programming」（M. I. S）がガンバッテいます．ところで，X Windowの開発元であるM. I. Tのマニュアルを翻訳出版した「創夢」という会社がありますが，現在全7巻のうちVol. 0，1，2，3，6巻が刊行されており，興味深いものがあります．

最近目を引いたのは，「オブジェクト指向入門」（アスキー）で，これはP. H社刊Meyer著「Object－Oriented Software Construction」の訳本です．Meyer氏はオブジェクト指向言語として評判の高い「Eiffel」を開発した研究者です．近日同社より「Eiffel:The Language」「Eiffel:The Environment」「Eiffel:Library」の邦訳出版が予定されています．

■和書&洋書ベスト10　（書泉グランデ調べ：1990年10月1日～11月11日）

1	LaTex	アスキー	1	UNIX Network Programing	Prentice Hall
2	X Window OSF/Motifプログラミング	日刊工業新聞社	2	Res Edit Reference	Addison Wesley
3	X Viewプログラミング入門	スペック/工学図書	3	Compilers (paper)	Addison Wesley
4	C言語を256倍使うための本	アスキー	4	Macintosh C Programming primer V2	Addison Wesley
5	VZを256倍使うための本	アスキー	5	Advanced X Window Application programming	M. I. S
6	入門編Ver.3	エーアイ	6	Internetworking with TCP/IP V1 2/F	Prentice Hall
7	プログラミング言語AWK	トッパン	7	Macintosh C programming primer 1	Addison Wesley
8	プロフェッショナルのパソコン利用術	翔泳社	6	Annotated C++ Reference Manual	Addison Wesley
9	健康ソフトハウス物語	ラディク／星雲社	9	Turbo C++ Disk Tutor	McGrow Hill
10	プログラミング言語C 第2版	共立出版	10	Inside Smalltalk V2	Prentice Hall

米国ではWindows4.0の計画が発表されている．Windows4.0はOS/2のPMの機能を盛り込み，将来的にはOS/2上でWindowsを動作させるとのこと．

Programmer's Program

ソースファイルの履歴を管理する開発支援ツール

RCS(アールシーエス)

ソースファイルの差分を記録して管理

RCSは,「Revision Control System」の略で「リビジョン(Revision=改訂)管理ツール」のことである.もともと,RCSはソースファイルのバージョンアップ履歴を管理するためにUNIX BSD版上で開発されたツールで,UNIX System VにもSCCSという同等のユーティリティが付属している.

ソースファイルのアップデートの差分を記録することによって,すべてのバージョンアップされたプログラムのソース全体を保存するよりも,はるかに少ないファイルサイズで,すべてのリビジョンを残すことができる.しかも任意のリビジョンをいつでも復元できる.このUNIX上のRCSの機能の多くをMS-DOS環境下で実現したプログラムがオンラインソフトウェアとして流通しているが,今回ここで取り上げたのは,日経MIXのlong2会議室にISHファイルで登録されているT. Tanaka氏(日経MIX:PSH)が製作したRSCである.

使用環境は正しく設定しておく

RCSは,FDDベースでも使用できるが,かなり大きな作業領域を必要するので,ハードディスクでの利用が前提となる.まず,適当なディレクトリでRCS.LZHを解凍する.RCSのコマンドは,ほかのコマンドを子プロセスとして起動するので,各コマンドを起動できるように,格納したディレクトリにあらかじめパスを通しておく必要がある.

次に,環境変数を設定する.環境変数USERは,リビジョンを登録したり,変更のためのロックをかけたりするときに参照される.

そして,ソースを格納するディレクトリにRCSというサブディレクトリを作成すると準備が完了する.

どのリビジョンでも復元することが可能

ソースファイルの登録や更新は

A> CI <file Name>

と入力する.ここでコメントを要求してくる.このコメントは登録時にはファイルの説明などを,更新時には更新内容を入力する.コメントの入力終了は文頭で「.」と入力する.なお,ここで指定したソースファイルは削除され,RCSディレクトリにリビジョン履歴データとともに格納される.

最新のリビジョンの復元は

A> CO -l <file Name>

と入力する.これでカレントディレクトリに最新のリビジョンナンバーを持つソースファイルが復元される.また,いろいろなオプションを付加することによって,任意のリビジョンナンバーをもつソースファイルの復元などが可能.ほかにも,アップデートの履歴をみるRLOGや,変更点をみるRCS DIFFなどのコマンドが用意されている.

リビジョン履歴の出力例

```
type    a;
owner   Yoshiyama;
head    1.2;
access  ;
symbols ;
locks   ;
mode    strict;backup;
leader  ; * ;
total   2;
comment 画面上に"Hello"を表示するプログラム
;
1.1;
seq 1;
date    90/11/21 23:07:59;
author  Yosbiyama;
state   Exp;
mother  0;
comment Initial revision
;
1.2;
seq 2;
date    90/11/28 20:56:57;
author  Yoshiyama;
state   Exp;
mother  1;
comment 画面表示を"Hello wold"に変更
;
s
i 1 /*  RCS出力結果確認用TEST.C     */
i 1
i 1 main(){
i 1
i 1     printf("Hello¥n");
d 2
i 2     printf("Hello world¥n");
i 1
i 1 }
e
```

●RCS

登録名:RCS.LZH/著作権者:T. Tanaka(日経MIX PSH)/動作環境:MS-DOS汎用

ワンラインNEWS
松下電器産業はWSの最大手メーカーである米サンマイクロシステムズ社から,サン互換WSの開発のためのライセンス受けることを合意したと発表した.

プログラマ登場②

FORTHもFIFTHもMindも，ほしいからつくった

株式会社リギーコーポレーション
片桐　明

ハードもソフトも一緒

片桐氏は，あるときにはハード屋であり，あるときにはソフト屋であり，またビジネスを思考する経営者でもある．「私にはハードもソフトも区別がないんです」というように，仕事の経歴をみても，その時々に興味を持ったことをやり続けてきたという印象が強い．

最初の会社ではミニコンのソフト開発，次の会社ではトランシーバー関連のハード製作などを行った．3回目の転職先ではマイコンTK-80のボードの製作などハード関連からやりはじめたが，そのうちTK-80のソフトも作るようになった．当時は，マイコンというとTK-80ぐらいしかなかったので，そのうち自分たちで部品を買ってきてIDS-8000というマイコンをつくって販売しはじめた．それと同時にIDS-8000のOSの開発を手がけるようになったのが，片桐氏がOS開発を手がけた最初だった．「当時は自分でほしいと思ったものは，自分で作るというのがあたりまえでしたからね．この考えは今でも同じで，私はFORTHもFIFTHもMindもすべて自分でほしいと思ったからつくったんです」．

片桐　明（かたぎりあきら）
昭和25年長野県に生まれる．東京電気大学電子工学科卒．日本語プログラミング言語Mindの開発者．株式会社リギーコーポレーション代表取締役社長．著書に『日本語プログラミング言語Mind』（翔泳社）がある．

日本人の発想のままに

片桐氏を高級言語の開発に奮い立たせたものは，スタック指向言語「FORTH」との出会いであった．「スタックというハードで用いられている手法を，コンピュータソフトに最初に採用したチャールズ・ムア（FORTH開発者）の発想のすばらしさに感動した」という．ここからが片桐氏のすごいところである．FORTHを手にいれようとは少しも考えず，1日だけのFORTHセミナーでもらった資料をもとに，自らFORTHの開発に取り組んだ．意外にもこの言語は単純で，わずか1か月で完成してしまった．

次に開発した言語FIFTHが，片桐氏のはじめての日本語プログラミング言語である．日本人がプログラミングをするときに，一度頭の中で日本語の構造で考えたものを，プログラムとして記述するときに別の構造に置き換えて記述しなければならないことは，非常に無駄だと考えたのが，FIFTH誕生のきっかけだ．

それがMindへと発展していった契機には，Turbo Pascalの出現がある．Turbo Pascalによって，大衆向け高級言語の可能性に開眼した．従来の言語より構文などを大幅に単純化し，FIFTHの日本語ノウハウにさらに磨きをかけ，100%日本語でプログラミングする言語としてMindを開発した．Mindは開発中にTurbo FORTHというコードネームでよばれていたという．Turbp Pascalから受けた影響のほどがうかがわれる．

●日本語プログラミング言語Mind

Mindは，使用する言葉が日本語であるだけでなく，句読点や丸の使い方，言い回しを含めた文法がほとんど日本語になっている．プリプロセッサのようなものではなく，直接機械語を生成するコンパイラである．「スタック」という仕掛を導入することにより，構文解釈・意味解釈といった手法を使わずに日本語表現のプログラムを解釈・実行する．スタックは，計測・制御・グラフィック処理などで多く使われている言語FORTHなどで採用されている手法．

製品名：日本語プログラミング言語Mind
定価：39,800円
問い合わせ先：㈱マイクロソフトウェアアソシエイツ　TEL 03-3486-1411

オンラインソフト採集帳 2

一度使うと手ばなせない便利なファイル管理ツール FD Ver2.21

ハードディスクは，いちどに多くのファイルを格納できるという大きなメリットがある反面，しっかり管理しないと迷路のようなディレクトリと，迷子になった無数のファイルの山で大切なデータが埋もれてしまうというデメリットもある．今回紹介するFDは，ハードディスクなどのファイルを管理するオンラインソフトである．FDを使うと，MS-DOSのプロンプトでフルパスを指定しながらファイルのコピーや移動をする気はおきないだろう．すでに市販のファイル管理ユーティリティを使っている方にも，ぜひ勧めたい必携ツールのひとつである．

まずエディタなどを登録し環境設定を行う

ハードディスクに適当なディレクトリを作り，そこへFD221.LZHを解凍する．圧縮ファイルからは8つのファイルが現れれる．まずFDCUSTを起動して，使用するエディタ・圧縮プログラムの登録と環境の設定をおこなう．そしてパスを設定してインストールは完了．ただし，FDの全ての機能を使うためには，ほかにも圧縮・解凍ユーティリティやエディタが必要なので，これらを別途用意しあらかじめパスを設定しておく必要がある．

準備ができたらDOSのコマンドラインから

```
A> FD
```

と入力してFDを起動する．

シンプルな画面から多様なファイル操作

FDの画面は，カレントディレクトリにあるファイルとその情報やドライブの残容量，日付と時間などで構成されている．いたってシンプルだが，必要な情報を全て表示している．FDはファンクションキーまたはキャラクターキーによって各機能を実行する．たとえば，内容を確認したいファイルがある場合，矢印キーを使って目的のファイルへカーソルを移動し，[f･9] または [V] を押す．同様に，[f･2] または[X]を押すとプログラムが実行できる．ほかにもファイルのコピー，削除，編集などが可能．さらに複数のファイルをスペースキーや [TAB] によって選択することもできる．ディレクトリにファイルがたくさんあり，目的のファイルが表示されていないときは，[ROLL UP]か[ROLL DOWN]を押せば次のページが表示される．また1画面のファイルの表示列を1列(18/画面)から5列(90/画面)まで，[1] [2] [3] [5] で変更することもできる．

アプリケーションを登録するとキー一発で起動できる

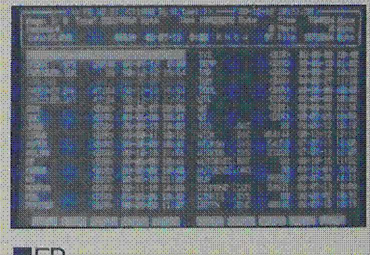
FDCUSTを使うことで，よく利用するアプリケーションを最大52個まで登録することができる．登録したアプリケーションの起動は [GRAPH]（または [XFER]，[NFER]）と任意の [A] から [Z] である．ここへ通信ソフトなどを登録しておけば簡単に起動ができる．これらの機能はほんの1例で，ほかにも優れた機能がたくさんある．機能について詳しくは，FDに付属のマニュアルで述べられている．

使ってみて感じるのは，ファイルの拡張子を判断して起動できるアプリケーションの種類をもっとふやしてほしいということだろうか．たとえば，表計算やカード型データベースを起動できるようにするなど，バージョンアップが期待される．便利なツールであるだけに，つい欲がでてしまう．

市販のファイル管理ユーティリティも，最近は多機能なものが次々と発売されてはいるが，FDほど本来の機能が充実したユーティリティは，なかなか見ることができない．

また現在作者の出射さんはIBMのPS/55用のFDを製作されているとのこと．本誌が発売される頃には各ネットにも出回っているだろう．

せっかくのハードディスクなので，しっかりとファイル管理をして気持ちよくプログラミングをしたいものだ．



■FD

分類:ファイル管理ユーティリティ/登録名:FD221.LZH/著作権者:A.Idei(出射　厚，NIFTY - Serve:MHB01504)/動作環境:PC9801シリーズおよび，その互換機/掲載月日：1990年11月11日/入手年月日：1990年11月22日/入手場所：Genesis(03-280-4571 24時間 ～9600bps(V.32/HST))/解凍方法：LHARC E FD221

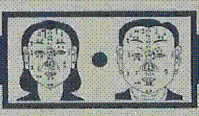
ワンラインNEWS
山口のソフトハウス「エムシーシー」は議員の後援会事務所向けの会員管理用ソフト「当確参謀」を開発した．来年の統一地方選挙にむけて後援会事務所に売り込む．

■機能および特徴

ファイルの内容表示・編集/複数のファイルのコピー・リネーム・削除・移動・圧縮・アトリビュートの変更・タイムスタンプの変更・圧縮/圧縮ファイルの解凍/プログラムの実行/ディレクトリの作成やディレクトリ中のファイルを含めた削除/ユーザーが任意に設定できる52個の外部コマンドの実行/1画面に最大90個のファイルを表示/コピーや移動時などのドライブやディレクトリ指定は直接指定とツリーで選択/ワイルドカードを利用してファイル表示/フロッピーディスクのフォーマット付きバックアップ/ファイルの内容表示中にエディタを起動でき，MIFESなどエディタであれば論理行へのジャンプ/[RET]でカーソル位置のファイルが圧縮ファイルなら圧縮されたファイルのリストを表示，実行ファイル(拡張子が，COMや，EXE)以外ならファイルの内容表示/[SHIFT]+[RET]でカーソル位置のファイルが実行ファイルなら実行，それ以外の拡張子ならエディタで編集/キーボードマクロ

ファイル構成

```
Listing of archive : 'FD221.LZH'

 Name          Original    Packed  Ratio   Date      Time    Attr Type  CRC
-------------  --------  --------  -----  --------  -------- ---- ----- ----
 FD.CFG            4074      1792  44.0%  90-11-11   2:21:00 a--w -lh1- 4474
 FD.COM            1738       731  42.1%  90-11-11   2:21:00 a--w -lh1- 27B8
 FD.DOC           46528     18819  40.4%  90-11-11   2:21:00 a--w -lh1- E029
 FD.EXE           33214     24476  73.7%  90-11-11   2:21:00 a--w -lh1- BEAF
 FD221.DOC         1207       772  64.0%  90-11-11   2:21:00 a--w -lh1- C2B0
 FDCUST.COM        5856      3059  52.2%  90-07-30   2:20:00 a--w -lh1- BACA
 FDCUST.DOC        8753      3713  42.4%  90-07-30   2:20:00 a--w -lh1- 9409
 FDCUST2.COM       1555      1274  81.9%  90-11-11   2:21:00 a--w -lh1- F6E9
-------------  --------  --------  -----  --------  --------
    8 files      102925     54636  53.1%  90-11-11  17:41:20
```

HEALTHY?  UNHEALTHY?

② プログラマもカゼをひく カゼをひいたら……

このあいだK氏はせきをしていた，Nさんは熱で会社を休んだ，などという例をあげるまでもなく，プログラマもカゼをひく．カゼといっても症状はさまざまで，ひきはじめに症状にあった薬を選んで飲むとうまい具合になおってしまうことも多い．症状にあわせて選ぶことがポイントだ．ということで，今回はカゼ薬の種類を簡単に説明しよう．

■総合感冒薬

身体がだるい，頭が痛いなどの全身症状があるときに．例：コルゲンコーワホワイト顆粒，コンタック総合感冒薬（ビタミン剤入り）

■熱と鼻みずのカゼ薬

鼻みず，のどの痛み，熱はあるがせきはでないときに．カゼの初期に使うと効果的．例：パイロンAカプセル

■用途別カゼ薬

①眠くならないカゼ薬

抗ヒスタミンが入っていないので，眠くならない．車の運転，仕事をしなくてはならないときに便利．ただし，鼻カゼではなく，熱とせきの症状向き．例：パブロンアウェイク，改源

②胃にやさしい風邪薬

胃の弱い人のために，胃粘膜を保護する成分をプラス．例：アルペンエースカプセル，アルペン錠

③漢方のカゼ薬

どうしても化学物質の合成薬はいやだというこだわり派向け．なかなか熱の下がらないカゼ，せきのでるカゼに効果的．例：カンポー（桂麻各半湯のエキス顆粒）

Computer Laws

コンピュータの法律

プログラム著作権とはどんなものか――その1

熊谷信太郎(弁護士)

今回は,プログラムは法的にどのようにプロテクトされているかを考えてみたいと思います.日本では,プログラムに代表されるコンピュータソフトは,著作権法によって法的な保護が与えられています.そこでまず,著作権とはどのようなものかについて,基礎的な説明をしたうえで,次回以降に,プログラムと著作権に関する具体的な問題を検討したいと思います.

■そもそも著作権とは

著作権という言葉は一般によく使われますが,法律的にいうと著作者人格権と狭義の著作権という2つの権利で構成されています.

著作者人格権というのは,著作者の人格的価値の保護を目的とした権利です.公表権,氏名表示権,同一性保持権などが著作者人格権を構成する要素です.まず,公表権とは,まだ発表されていない著作物を公衆に提供したり提示する権利のことです*1.次に,氏名表示権とは,著作物の原作品や著作物を公衆に提供・提示する際に,氏名を表示する,または表示しない権利です*2.同一性保持権とは,同意なしに,著作物の変更・切除その他の改変を受けない権利のことです*3.

著作者人格権は,その名の通り,著作者の人格と結びついた権利です.これは,ひとりの人に専属した権利であり,ほかの人などに譲渡することはできません*4.

これに対し狭義の著作権(以下,著作権とはこの意味で使います)とは,著作物の利用に関する財産的な利益を保護することを目的とした財産権であり,複製権をはじめ,さまざまな権利で構成されています*5.この著作権は,財産権ですから,著作者人格権と異なり,ほかの人などに譲渡することも自由です.

著作者人格権,著作権ともに,権利の侵害があった場合には,①差止め請求,②損害賠償請求,③名誉回復等の各措置をとることができます*6.

差止め請求は,著作権等を侵害している者,または侵害しようとしている者に対して,侵害の停止または予防を請求することです.著作者が死亡したあとでも著作者の遺族はこれらの措置をとることができます.

■著作権は無制限ではない

このように著作権法は,著作物に対する著作者の利益を保護する規定をおいていますが,著作者の利益を守るだけでなく,「文化的所産」を公正に利用することで「文化の発展に寄与すること」も目的に掲げています*7.

そのため,著作権も絶対無制限なものではなく,「著作物の公正な利用」や文化の発展のために,条件を決めて著作権を制限する規定が置かれています*8.その条件にあえば,著作権者の承諾を得なくても著作物を利用することができるのです.その条件は,私的目的で使用する場合と,公共的な目的で使用する場合とに大別されます.私的利用とは,個人的に,または家族内等で利用する場合です.これは,「文化の

Illustration
Nobuki Fukuhara

ワンラインNEWS
日本AMDは業界最速の256ビットEPROM(紫外線で消去,再書き込みの可能なROM)である「am27H256」を発表した.アクセスタイムは35ナノ秒をほこる.

発展に寄与する」という著作権法の立法趣旨からして，私的な使用を制限すべきでないということです．

これに対し，公共目的で使用する場合は，図書館や学校での使用や，裁判，報道のための使用など，多岐にわたっています．いうまでもなく，著作権の保護も絶対的なものではなく，著作物の公正な利用と著作権の保護との調和をはかるという趣旨で，これらの公共目的での使用を許しているのです．

■著作権の保護期間は50年

著作権は，著作者が個人の場合には，著作物を創作したときに権利が発生し，原則として著作者の死後50年間存続します*9．

これに対し，会社でプログラムを作成した場合のように，著作者が法人等の団体の場合には，著作物を公表してから50年間(創作時ではない点に注意)権利が存続します．ただし，創作後50年内に著作物が公表されなかった場合は，創作時から50年で権利が消滅します．

■プログラムは「著作物」である

著作権が認められるためには，「著作物」である必要があります*10．コンピュータプログラムが著作権法上の「著作物」であるかについては，かつては議論があり，それは「著作物」というより，むしろコンピュータという機械を動かす「技術」であるから，特許などの工業所有権の一種として保護すべきであるという立場からの反対論もありました．とくに，特許の場合には，保護される期間は15年ですが，著作物は前述のように50年間保護されるので，プログラムを著作物とみると，技術の発展を阻害するという批判が強かったのです．

しかし，プログラムを著作物とみて，著作権法で厚く保護するのは，世界的な趨勢であり，日本でもそれにならって，昭和60年の著作権法改正により，プログラムは著作権が認められる「著作物」であると明文化しました*11．

著作権はおおむねこのような内容の権利ですが，次回は，プログラム著作権に関する具体的問題を検討したいと思います．

*1　著作権法第18条
*2　著作権法第19条
*3　著作権法第20条
*4　著作権法第59条
*5　具体的には，次のような権利で構成されています．
①複製権：著作物を複製する権利であり，著作権の本質をなす権利です．(著作権法第21条)
②上映権：(著作権法第22条)
③放送権，有線送信権等(著作権法第23条)
④口述権：言語の著作物を公に口述する権利です．(著作権法第24条)
⑤展示権：美術や写真の著作物を公に展示する権利です．(著作権法第25条)
⑥上映権，および頒布権：映画の著作物を公に上映，頒布する権利です．(著作権法第26条)
⑦貸与権：(著作権法第26条の2)
⑧翻訳，翻案権：(著作権法第27条)
この中には，たとえば展示権のように権利の性質上，美術や写真の著作物についてのみ認められ，プログラムの著作権には含まれないものもあるので，すべての著作物について，これらの権利が当然に認められるわけではありません．
*6　著作権法第112条ないし第115条
*7　著作権等の文化的所産の公正な利用を図り，もって文化の発展に寄与すること．(著作権法第1条)
*8　著作権法第30条ないし第47条の2
*9　著作権法第51条
*10　著作物とは，思想又は感情を創作的に表現したものであって，文芸，学術，美術又は音楽の範囲に属するものをいう．(著作権法第2条)
*11　著作権法第10条第1項第9号

アキバ短信

営業時間9:30～22:30(平日) 9:30～21:30(日祝) 年中無休
秋葉原電波会館2階　TEL 03-3253-1746

古炉奈
迷宮のオアシス

家電から電子デバイスにいたるまで秋葉原は何でもそろう電気のデパートである．しかしこの巨大なデパートには最上階がない．デパートの最上階にあるべきもの，飲食店の数が極端に少ないのである．そこで秋葉原電気街にきた買物客の多くはお茶も飲まずにそそくさと電車に乗り込んで家路を急ぐ．

だが，秋葉原にも憩いのスポットはある．歩き疲れてちょっとひと休みというときに便利な，電波会館2Fの「古炉奈」を紹介しよう．秋葉原デパートに隣接した高架下電気街の旧神田市場側にある．以前は「コロナ」という名で全国どこにでもあるような場末の喫茶店だったが，改装してイメージ一新，落ち着いた雰囲気の中でゆったりコーヒーを飲める店に変身した(と同時にコーヒーの値段もアップしたのだが)．この変身は，アキバに集まるマニアの間でもひとしきり話題になったものだ．

というと，まるでむさっくるしい男ばかりがいるようだが，そんなことはない．ちゃんと妙齢の女性の姿も見かける．乞うご期待．

IBMは通信関連ビジネスの本部を米国から欧州に移す準備を始めた．これは欧州には通信の標準化団体が数多く存在しているためといわれている．

Vertual Space Guide

Illustration : TOSHIYA SHIROTANI

バーチャルスペースガイド

新宿の寺院建築

P子　どうしたの目が真っ赤じゃない.

C平　正月にホテルで目が赤くなるようなことしちゃったから.

P子　なによ，おもわせぶりに……．どうせエッチ関係じゃないことはわかってるんだから．で，何したの.

C平　新宿のホテルから新しい都庁を見てた.

P子　なーに，それ.

C平　だって，あの建物は雰囲気が怪しいじゃない．それで，新宿のあのあたりが全体的にサイバーパンク化するんじゃないかと思って，写真とったりしてるうちに，窓の数かぞえはじめて……それで，目が疲れたみたい.

P子　窓の数かぞえたのはバカとしかいいようがないけど，たしかにあの建物は怪しいよね．ゴシックの尖塔だかなんだかしらないけど，中世だよ．映画のセットならわかるけど，よくあんなものつくったよね.

C平　そう，あれは構造として寺院建築だよね，都議会の建物から尖塔にかけて大きな回廊があるけど，回廊と塔という組み合わせはまさに寺院建築の典型だもの．梅雨時になって雨にかすむ様子は本当にサイバーパンクだろうと思う.

P子　それを，今からたのしみにしてるわけ．何考えてるんだろうね．でも，気持ちはわからないでもないよ．梅雨時に西口の鯨カツ屋あたりでビール飲みながら表を見ると，雨にネオンがにじんじゃって……まあ，あれはサイバーパンクだよ.

でも，『ブレードランナー』のリドリー・スコット監督が撮った『ブラック・レイン』はちょっと期待はずれだったわ．日本人にとってみればそれほどサイバーパンクじゃないのよネ．外人が日本を撮るときのヘンテコさが鼻についた．クライマックスの舞台なんか，日本だかマカロニウエスタンだかわからない感じだったし，かといって国籍不明なところがおもしろいわけでもない．映画としては，まあおもしろかったと思うけど，『ブレードランナー』の状況設定がすごかっただけに，ちょっと残念.

C平　それはいえる.『ブレードランナー』は，原作の『アンドロイドは電気羊の夢を見るか』とはぜんぜん違う独自の世界を築いていたもの．ハリソン・フォードが演じたデッカード刑事も一種のピエロなんだけど，原作の刑事はもっと間抜けだし，だいいち雨が降っていない.

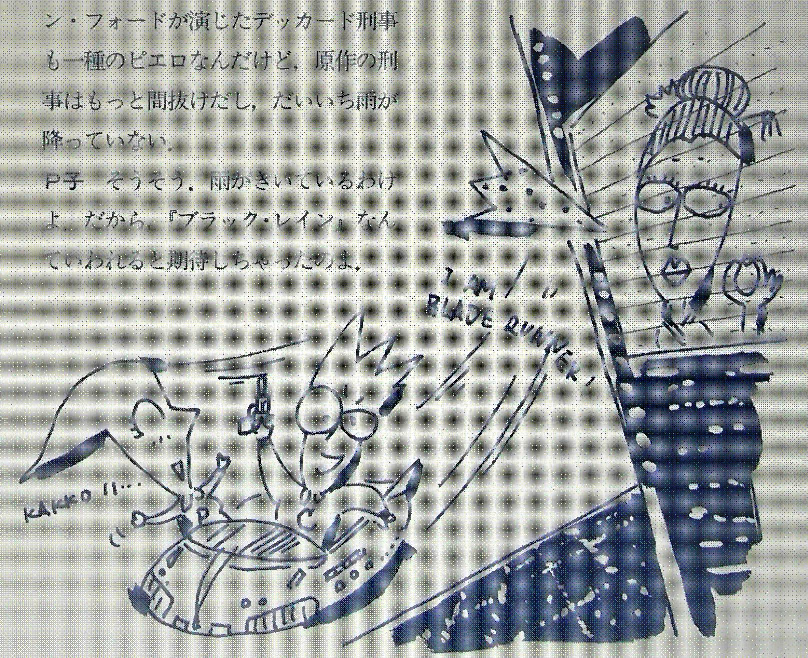
P子　そうそう．雨がきいているわけよ．だから，『ブラック・レイン』なんていわれると期待しちゃったのよ.

バラードをマルチメディアで

C平　フィリップ・K・ディックの小説は，それほど視覚的なイメージは浮かばないけれど，かえってそれが映画の成功に結びついているのかな．映画の演出をするほうでやりやすいのかもしれない．そこにいくとかつてSFニューウェーブといわれたＪ・Ｇ・バラードの小説は，すごく視覚的だけど映画化されていないね.

P子　うーん．『太陽の帝国』はなかなかの傑作だったけど，バラードでもあれはSFじゃないものね.『太陽の帝国』は，原作がいいよ．最後のほうで原爆が出てきて，その描写から感じられる歴史観みたいなものにバラードらしさがでている.

C平　あ，原作はまだ読んでない．そ



P子　某ソフトハウスで企画営業を担当している。26歳。
C平　某メーカーのシステムソフトウェア開発部門に所属。25歳。

れも貸して欲しいな。「ちびまる子」はまだ返してもらってないけど……。

P子　このあいだから，ちびまる子，ちびまる子ってうるさいね，ちゃんと返すよ。でも，もしかして『結晶世界』あたりが映画化されるといいなんて思っているわけ。多分あれの映画化はたいへんよ。アフリカが結晶になっちゃうんだから。

C平　SFとしては，『結晶世界』も好きなんだけど，映像化するといいと思うのは『ヴァーミリオンサンズ』だね。宝石をちりばめた昆虫が出てきたり，雲の彫刻家がいたり，砂エイが砂漠を飛んできたり……すごい映画になると思うけどね。

P子　はいはい。「プリマ・ベラドンナ」や「スクリーン・ゲーム」といった短編が集められているヴァーミリオンサンズ・シリーズね。あれを読んでいると頭のなかで景色を思い浮かべてしまうわ。

C平　欲をいえば，インタラクティブなマルチメディアみたいなものになるといいな。ヴァーミリオンサンズというバーチャルスペースがあって，それをCD-ROMにいれてさ，分岐をたどっていくとそれぞれの短編のお話が展開されるみたいな……。

P子　まあ，夢を見るのは勝手ですけど，それにはマルチメディアプレーヤーみたいなもんがいるんでないの。まだ，ないんだよそんなもんは。

C平　それが問題なんだよな。デモディスクみたいなものになったんじゃつまんないし，実現できれば本当のニューメディア作品になるんだけど……。

DATA

■ブレードランナー(1982年・米)

監督：リドリー・スコット
出演：ハリソン・フォード

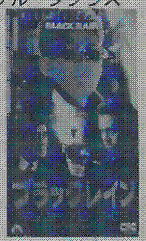
無菌室のような清潔なイメージで描かれることの多かった未来社会を，猥雑な混沌として視覚化して，いまも熱心なファンをつかんでいるSF映画。2020年のロサンゼルスを舞台に，反乱をおこしたレプリカントをハリソン・フォードが追う。サイバーパンクの記念碑的作品だ。原作は，『アンドロイドは電気羊の夢を見るか』(ハヤカワSF文庫／フィリップ・K・ディック)。

13,900円　ワーナー・ホーム・ビデオ

■ブラック・レイン(1989年・米)

監督：リドリー・スコット
出演：マイケル・ダグラス

ヤクザを追って大阪にきたアメリカ人刑事が日本の警察の捜査制限にいらだちながらもヤクザ組織と対決する。最初は米国人刑事の見張り役で，最後は行動をともにする日本の刑事役に高倉健，冷酷なヤクザ役にこれが最後の映画となった松田優作が扮する。
刑事とヤクザがからむオーソドックスなアクション映画であるが，出演者はそれぞれのもち味を出している。大阪周辺のロケ映像も見もの。



14,830円　CIC・ビクター／AVC

■太陽の帝国(1987・米)

監督：スティーブン・スピルバーグ
出演：クリスチャン・ベール

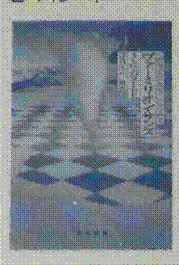
上海に住む14歳のイギリス人少年が第２次世界大戦の戦火に巻き込まれながら，生きながらえ，成長していく姿が描かれる。飛行機に対するあこがれ，戦争の非情さなどが少年の目でとらえられて，新鮮な感動を与えてくれる異色戦争映画。原作は，『結晶世界』『時の声』などのSF小説で知られる英国の作家J・G・バラードの『太陽の帝国』(国書刊行会／J・G・バラード)。

14,800円　ワーナー・ホーム・ビデオ

■ヴァーミリオン・サンズ(ハヤカワ文庫)

著者：J・G・バラード

砂漠地帯にある架空の町ヴァーミリオンサンズを舞台にしたバラードの短編集。終末感を漂わせるバラードの作品のなかでも，ヴァーミリオンサンズシリーズは，ことのほか装飾的な退廃感が満ちている。バラードのほかの作品で終末を生きるのは，科学者であったり医者であったりするのに対して，ヴァーミリオンサンズの住民は雲の彫刻家やオペラ歌手などの芸術家である。



創刊記念プレゼント

毎月HANDY98を1名様にプレゼント!

創刊号から創刊第3号で実施!
(商品は月により多少異なります.)

1 HANDY98 (PC-98HA)

1名

2 Turbo C++ Professional

1名

ボーランド・ジャパン(株)提供

3 Microsoft QuickC

1名

マイクロソフト(株)提供

7 dBXL マグカップ

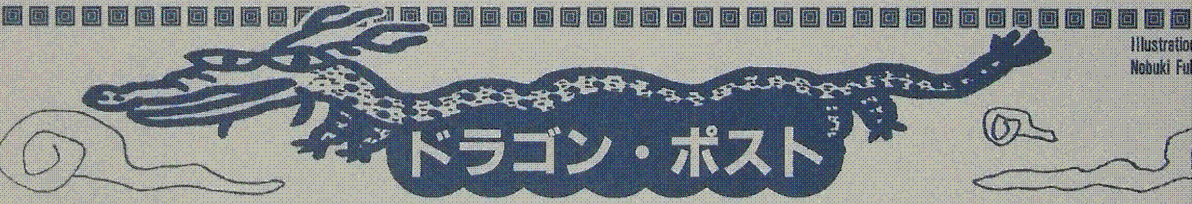
3名

サザンパシフィック(株)提供

11 OOPツアーTシャツ

3名

ボーランド・ジャパン(株)提供



ドラゴン・ポスト

Illustration Nobuki Fu...

この欄は，読者からのお便りを中心に構成します．本誌に対するご意見・ご感想などをお待ちしております．鬼より恐い読者からの声ということで，「ドラゴン・ポスト」です．本誌についているアンケートはがきにご意見の欄を設けてありますから，そこに書いてください．ペンネームでも歓迎です．ただし，はがきには本名と住所をしっかり書いてくださいね．掲載誌をお届けします．

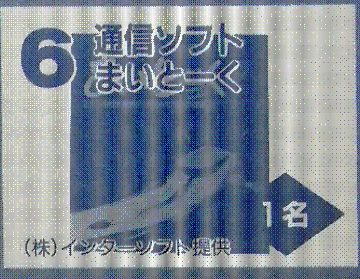
★本当に出た!?

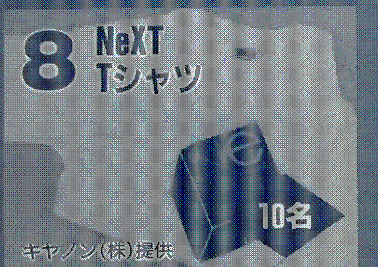
「プログラマーズ・ページ」という雑誌を出すということで，翔泳社からアンケートの依頼がきたのは，まだスキーができるころのことでした．秋頃から，いつ出るのだろうと思って書店にいくたびに気にはしていたのですが，なかなか見つけることができませんでした．「あれはまぼろしのアンケートだったのか」，そう思っていたときにあったではありませんか，「ザ・ベーシック」にプログラマーズ・ページの広告が……．12月18日創刊だって，やはりあれはまぼろしではなかった．と思ったのだが，ある時ネットの友だちとメールのやりとりをしていると，そいつが変なことをいうのだった．「プログラマーズ・ページってやっぱり出ないらしいよ」って，そこでまたぼくの心は揺れたのだが，12月18日になってすべては氷解した．だって，本屋さんで買ったのだから．(神戸市・SUIVI)

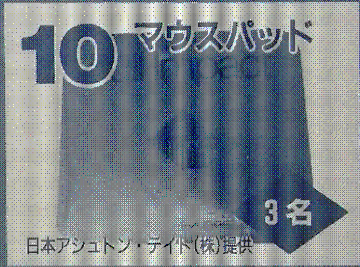
——編集部内でも出ないという噂がしきりでした (←こらこら)．^_^;

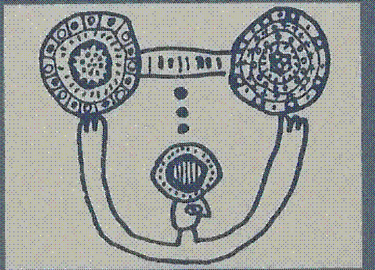
★アマグラマからプログラマに

ソフトウェアも人海戦術で作る時代です．この本を読んだ皆さんがアマグラマからプログラマになって人手不足解消になればいいですね．ところで，コーディングするのはSE









応募は本誌に添付のアンケートはがきをご利用ください（1991年 2月12日到着分有効）。なお，応募者多数の場合は，抽選になります．当選者発表は発送をもって代えさせていただきます．

の仕事ですが，プログラマの仕事はキーパンチャとどこが違うのでしょうか．求人広告で「初心者可」の文字を見るたびに考えてしまいます．やっぱりプログラマは一種の肉体労働でしょうか．ところで掲載されるとテレカかなにか頂けるのでしょうか．（所沢市・吉田）

――ええ，何もさしあげないというわけにはいきません．掲載号＋オリジナルテレカというせんでいかがでしょう．

★技術記事が少ない／

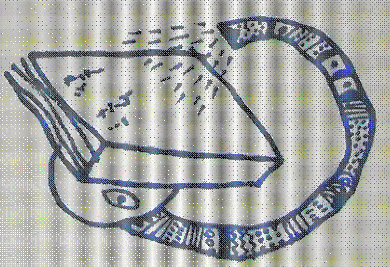
誌名の“プログラマ”の文字と“パソコンプログラマのためのクリエイティブマガジン”の言葉に誘われて購入しました．一読しての感想は，「技術的な記事が少ない／」というものです．ソースが出てくる記事がたった 2 つしかない雑誌に“クリエイティブ”の文字を冠するのは，どんなものでしょう？

特集の『プログラマの仕事場』にしても，プログラマが勤めている会社の概要，プログラミング環境と使用しているツールの紹介が主になっていますが，もう一歩つっこんで，どのようにその環境やツールを使いこなしているのかというノウハウまで紹介されていれば，ただの読物ではない実用的な記事になったと思います．

ちょっときついことを書きましたが，プログラマのための専門誌ということでおおいに期待しています．これからもがんばってください．

（静岡県　プータNo.1）

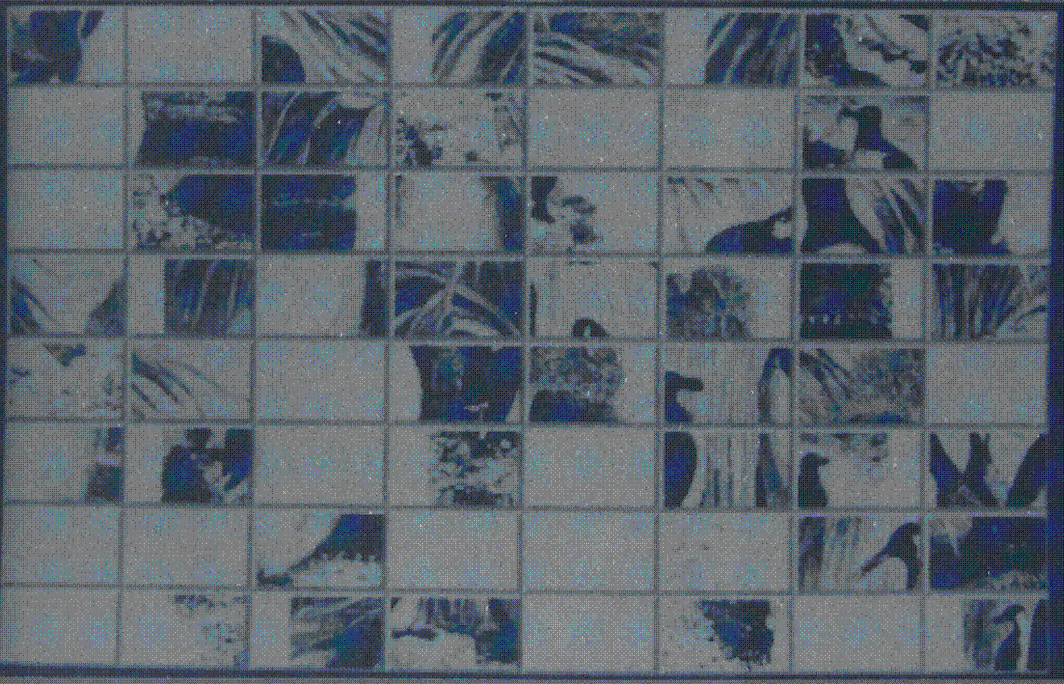
――クリエイティブなプログラマはソースの数にこだわったりしません．なーんちゃって．このような貴重なご意見がよりよい紙面を作りあげていくのですね(^_^;)．これからもよろしく．



クイズ・ランダムスクリーン

ある１枚の画像を64分割し，ランダムに並べ替えてあります．
さて，もとの画像は何を写したものでしょうか．

ヒント：南極の皇帝も仲間だよ…



正解者の中から30名様に本誌表紙イラスト（日暮修一画）入り月刊プログラマーズ・ページ特製オリジナルマグカップをさしあげます．

応募方法：本誌に添付のアンケートはかきをご利用ください．

締め切り：1991年２月12日（到着分有効）

なお，当選者の発表は発送をもって代えさせていただきます．

1月号の問題と正解

【問題】

【正解：福助】

月刊プログラマーズ・ページ
プログラムサービスのお知らせ

月刊プログラマーズ・ページでは，本誌に掲載したプログラムのサービスを行っています．
入手方法には，次の２つの方法があります．

1・BBSでのダウンロード

1. 03-5276-1297に電話をかけます．
2. 接続後IDの入力を求めてきますので，GUESTと入力し，リターンキーを押してください．
3. お知らせを表示後，メインメニューのコマンドプロンプトが表示されます．
4. Tと入力するとファイル転送システムに移行します．
5. ダウンロードする目的のプログラムがあるボードの番号を入力します
（＊と入力するとボードの一覧が表示されます）．
6. Bと入力するとファイルの一覧が表示されます．
7. ダウンロードするファイルの番号を入力します．
8. ご使用の通信ソフトにあわせて転送プロトコルを番号で選択します
（?でプロトコルの一覧が表示されます）．
9. ファイルの転送が終了したら，Oと入力し回線を切断します．

2・直接購入

希望メディアサイズ（5インチまたは3.5インチ），プログラムサービス希望の本誌号数をご記入のうえ，現金書留か郵便振替(東京4-154493)，または料金分の切手を同封のうえ，お申し込みください．

●送り先：〒102　東京都千代田区平河町2-4-14
平河町KSビル
株式会社　翔泳社
月刊プログラマーズ・ページ　プログラムサービス係
●価　格：1,500円（消費税，送料，ディスク代込）

・本サービスは入力の手間を省くため，本誌に掲載したソースファイル，実行ファイルを提供するものです．基本的には，掲載時のままでサービスいたしますので，プログラムのなかには，ライブラリファイルのように単独では実行できないものもあります．

＊画像データは，BBSでもサービスしています．

・掲載後に発見されたバグなどについては，本誌でお知らせします．また，マニュアルなどは付属していませんので，該当記事をご参照ください．

・プログラムの動作環境，対応機種などについては該当記事でご確認ください．

Programmer's Page

MONTHLY MAGAZINE FOR PROGRAMMERS

月刊プログラマーズ・ページ3月号
（2月18日発売）

特集

プログラマの主張
この言語で組め！

**いまやC一色に塗りつぶされたプログラミング言語．
ところが，こだわりをもってC以外の言語でプログラム
を組んでいるプログラマも少なくはない．
なぜ，Cなのか．
なぜ，Cでないのか．
「みんなが使っているから…」以外の理由が見えてくる．**

月刊プログラマーズ・ページ　1991年2月号
1991年2月1日発行

発行所　株式会社翔泳社
東京都千代田区平河町2-4-14平河町KSビル4F・5F
TEL 03-3263-0447　〒102
発行人　長廻健太郎
編集長　清水　隆
副編集長　鈴木嘉平
編集スタッフ　杉野貴美子＋上田純美礼＋芳山充弘＋勝山まゆみ
デザインスタッフ　佐々木幹夫＋SE（翔泳社デザイン研究所）
広告スタッフ　相澤正夫＋澤本良知＋石崎健治
協力　永井知己＋根本裕三郎＋シグマCC（株）＋小林弘人＋徳田洋
写植　株式会社東巧社
印刷　凸版印刷株式会社
定価　980円（本体951円，税29円）

© 1991 SHOEISHA
本書の一部または全部を著作権法の定める範囲をこえ，無断で複写，転載，テープ化，データファイル化することを禁じます．
＊本書の内容に関するご質問は，ご面倒でも必ず文書にて編集部までお問い合わせください．

EDITOR'S PAGE

●Windows 3.0の話題は，システムアーキテクチャとかAPIなどというかたいものになりがちだが，このソフトウェアの魅力はもっと別なところにもありそうである．画面のバックグランドにビットマップファイルで自分の好きな絵柄をいれることができる，あるいは画面の表示色をカラーパレットから選んで設定することができる，など細かいところまでいきとどいたユーザーインターフェースが用意されている．こうした緻密さと遊び心の調和がWindows 3.0の魅力になっている．プログラムアイコンを変えてみるなどというたわいないことも，使う側にとってはなかなかたのしい遊びになるのだ．（TA）

●うちの55/Zには日本IBMのDOS/Vがインストールされているが，このDOSはじつに画期的なしろものだ．なんといってもこのDOS/Vを使うと，漢字ROMを積んでいないフツーのIBM互換機で日本語が使用できるようになるのである．しかも，英語版のソフトはそのままなんの問題もなく動作し，うまくすると英語版のソフトで日本語の入力が可能になる．これで，ますますアメリカンソフトウェアに染まる毎日が続きそうだ．（嘉平）

●創刊の忙しさに，夫の誕生日を忘れてヒンシュクを買い，自分の誕生日さえも怒濤の如く過ぎ去り，気がつけばもう2号．そして90年も終わる．[慢性健忘症（歳のせいかな）＋体力の衰え（これも歳のせいだろうか）]にめげつつ，20代最後の年を迎えるにあたり，なにか目標をもった1年にしたいと思う．とりあえず，今年の目標は，引越＋運転免許＋海外旅行＋週に1度の定時帰宅（物欲的でせこい目標だな…）でしょうか．（ま）

●90年はひとり南の島へ旅に出た．91年こそは2人で……と毎年新年には決意をするのだけれどなかなか思うようには運ばない．田舎のお父さんお母さんごめんなさい．私は今年もひとりで残業してしっかり稼いで遊びに行きます．（kos）

●真夏に髪を切って涼しくとらばーゆしたと思ったら，あっという間に本格的な冬の季節を迎えてしまった．マフラーがわりのものがなくなってしまったので，今年の冬は寒さがジンジン身にしみる．寒さが大嫌いな私の唯一の冬の楽しみはやっぱりスキー．この号がでる頃には，我が編集部も余裕で締切を迎えられるような状況に変わり，優雅にスキー休暇をとれるようになっていることを切に希望しているのだが……．（K）

●今年は暖冬でどこも雪がないお蔭で，西武園にある狭山人工スキー場は大人気らしい．全国をさがしても「笑いの止まらないスキー場」はここくらいのものだろう．しかし，かき氷の上でのスキーも一興かも知れないが，早くパウダースノーの上でシュプールを描きたいものである．（よ）

愛読者カードご記入のお願い

本誌をお買い上げいただき、誠にありがとうございます。
お手数ですが、今後の誌面づくりの参考にさせていただきたいと存じますので、
各欄にご記入の上、ご返送くださるようお願い申しあげます。

月刊プログラマーズ・ページ READER'S SURVERY CARD 愛読者カード

本誌の号数　　　　年　　　　月号

本誌を知った理由
- □1.書店店頭　□2.他社出版物（　　　　　　）
- □3.小社出版物　□4. DM　□5.口コミ

パソコンの所有形態（機種名）
- □1.個人所有　（　　　　　　）
- □2.職場にある（　　　　　　）
- □3.持っていない

よく使用するソフトウェア

よく使用するOS

今月号でおもしろかった記事（いくつでも）

今後取り上げてもらいたい特集のテーマ

貴方が選ぶベスト3

パソコン雑誌	①	②	③
ソフトウェア	①	②	③
パソコン本体	①	②	③

本誌についてのご感想、お気づきの点、その他どのようなことでも結構ですので、ご意見をお聞かせください（「ドラゴン・ポスト＝読者の声」欄に掲載させていただく場合もあります。ペンネームをご希望の方はあわせてご記入ください）。

月刊プログラマーズ・ページ 広告資料請求カード

8	会社（学校）名		9	所属（学部）		10	従業員数	

※下記の該当する番号をご記入ください。

11	業種		12	職種		13	パソコン利用状況	

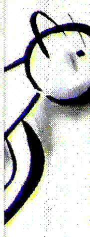
業種　①製造業　②金融・保険・不動産業　③商社・卸・小売業　④建設業　⑤通信・運輸　⑥システム・ソフト　⑦その他サービス業　⑧公務員　⑨その他

職種　①コンピュータ（ハード）技術者　②コンピュータ（ソフト）技術者　③エレクトロニクス技術者　④プログラマ　⑤その他の技術者　⑥専門職　⑦事務・営業職　⑧経営管理職　⑨商工サービス　⑩教職　⑪学生　⑫その他

パソコン利用状況　①会社で利用　②個人で利用　③会社でも個人でも利用　④利用していない

●希望される広告資料　　　年　　　月号

希望項目 ＼ 広告資料請求No.												
A 購入希望	A	A	A	A	A	A	A	A	A	A	A	A
B 説明希望	B	B	B	B	B	B	B	B	B	B	B	B
C 資料希望	C	C	C	C	C	C	C	C	C	C	C	C



定期購読のご案内

綴じ込みの振替用紙を使って、本誌の定期購読をお申し込みになれます。
1年間12冊＝11,760円（本体11,417円/税343円）
郵送料サービス

創刊記念 HANDY 98 プレゼント

郵便はがき

料金受取人払
麹町局承認
3908

差出有効期間
平成4年11月
30日まで

切手不要

102-00

東京都千代田区平河町2-4-14
平河町KSビル

株式会社 翔泳社
マーケティング部 行

月刊プログラマーズ・ページ READER'S SURVEY CARD 愛読者カード

氏名	性別 □1男 □2女	年齢 歳
住所 〒		電話 （　）　－

職業			ユーザー区分
□1ソフト技術者	□2ハード技術者	□3技術系会社員	□1プロ
□4事務系会社員	□5営業系会社員	□6教師	□2マニア
□7一般公務員	□8経営管理職	□9理系学生	□3一般ユーザー
□10文系学生	□11その他（　）		

創刊記念プレゼント記入欄	クイズ・ランダムスクリーン回答欄
希望商品番号	答

郵便はがき

料金受取人払
麹町局承認
3910

差出有効期間
平成4年11月
30日まで

切手不要

102-00

東京都千代田区平河町2-4-14
平河町KSビル

株式会社 翔泳社
月刊プログラマーズ・ページ
広告資料部 行

＊当社で記入

1	SE	2	受付No.

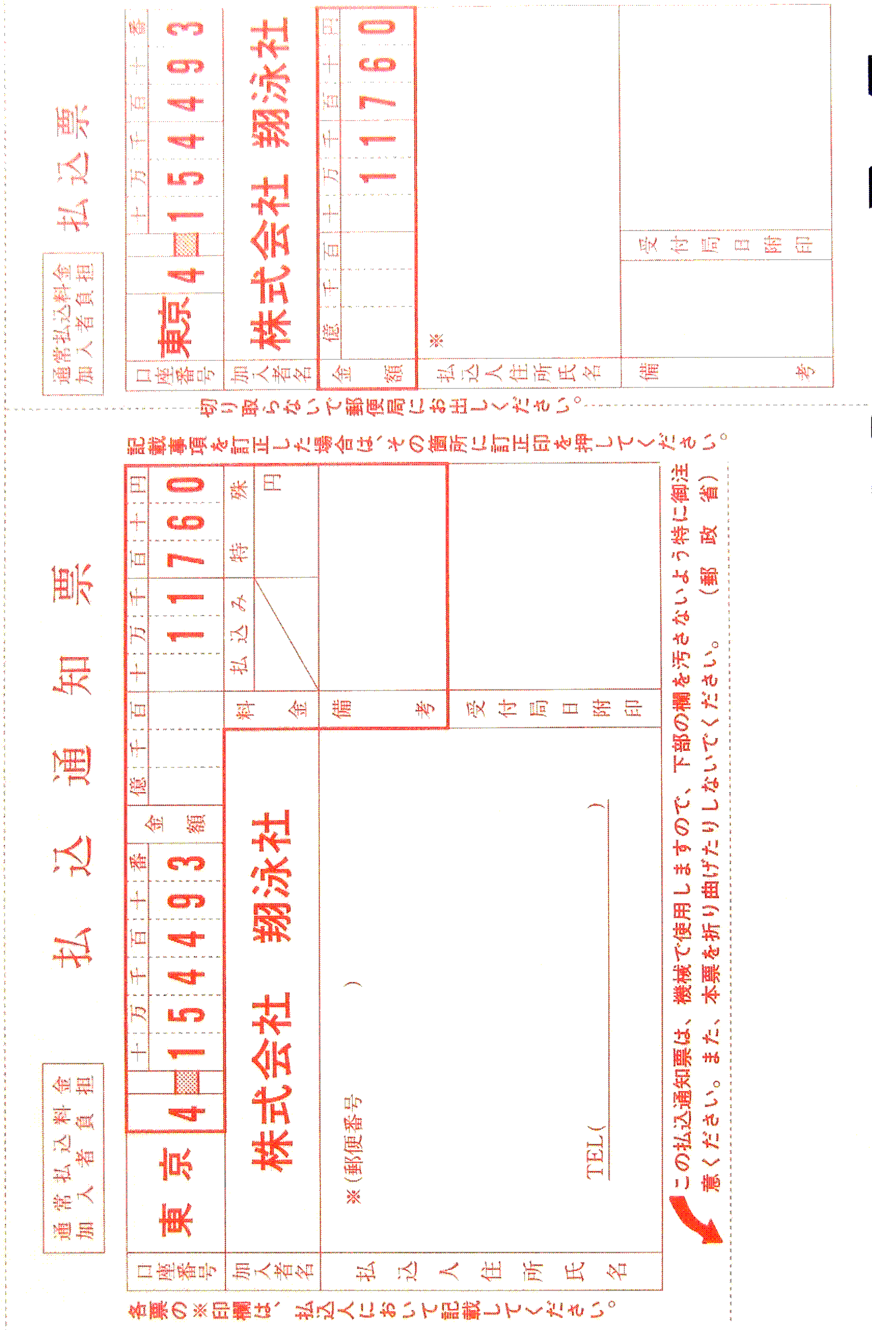
月刊プログラマーズ・ページ広告資料請求カード

3	会員番号	

＊すでに会員番号をお持ちの方はご記入ください。

4	フリガナ 氏名	姓　名	5	年齢 歳
6	自宅住所	〒　電話（　）		
7	会社住所	〒　電話（　）		

資料送付希望先の番号に○印をつけてください。



WE WANT OWER!

たくさんの駒が、自分の持ち分を踏まえながら頑張っているということだ。どの駒に属するかによって、業務内容から意識まで大幅に違う。では現場の人達は、どんな意識を持っているのだろうか。

●責任が顔に出る大手の技術者

「ちょっと大袈裟かもしれないけど、“自分達が日本を背負って立っているんだ”って気構えが心のどっかにありますね。極端な話、我々の間違いがもとで、日本中の金融業務がパニックになりかねないわけですから。」とは、銀行の通貨処理システムの企画・開発を担当するスタッフ談。正確に相手に伝えられないと、膨大な数の人間が無駄な作業に時間とお金を費やしてしまう。結構しんどいですよ。でも、重要な責務を担っていることに誇りがある。この仕事、気に入ってます。」と笑顔のB氏だった。

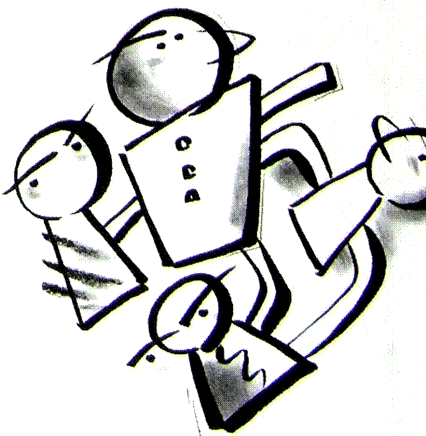
●実践は任せろ！—中小企業の精鋭部隊

さて、こうして大企業が受注したシステム開発を、実際に手掛けている現場、中小企業側の実態はどうなのだろう。

「請けた仕事を忠実にこなすのは、簡単なようで大変なんですよ。頭は使う、気は遣う。出向先で何をどうしたいか、

してほしいかなど、うまいことコミ
ケーション取って相手のニーズを理
ないと、仕事はスムーズにいきませ
とはC氏談。
の同期でいわばクビになったやつが
んだけど、彼すごく頭はよかったたん
でもどちらかというと研究者肌で、
閉じこもっちゃうタイプだったんで
で、取引先とのやり取りが全然出
くて、それイコール“仕事出来ない”
判断下されて終り。頭脳より対人処
力が必要なんです。」
の人の持っている技術より、人間性
要ということだ。
でも技術より対人処理能力とはいっ
資格は持ってたほうが有利ですよ。
って資格ある人とない人じゃ、ある
信頼できるでしょ？我々のような立
企業は、有資格者を前面に出して仕
請け負う傾向にあるから、いってみ
看板として、資格取っとくことをお
します。」

技術の最先端を追い求める人より、人と人との和、信頼を大切にしたいという人にはやり甲斐のある仕事に違いない。



定期購読のご案内

■定期購読は、1年間12冊と限定させていただきます。

■購読料のお支払は、この振替用紙をご利用ください。銀行振込、現金書留によるご送金は、ご遠慮ください。

■振替用紙に、ご住所、お名前をご記入のうえ、郵便局の貯金窓口にてお支払ください。

■小社より領収書の発行はいたしませんので、郵便局にて「郵便振替払込金受領証」をお受取ください。

■お申し込みいただいた方には、弊社より直接発送いたしますので、到着が発売日より遅れることがございますが、ご了承ください。

月刊プログラマーズ・ページ　定期購読直接申し込み書

この欄は、加入者あての通信にお使いください。

Programmer's Page MONTHLY MAGAZINE FOR PROGRAMMERS

年　　月号より

年間購読料 11,760 円(税・送料込み)

■ご希望の購読開始月号をご記入ください。

■送付先（住所・氏名は正確にご記入ください）

住所	□□□-□□	フリガナ	
		氏名	
		TEL	

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。　（郵　政　省）

郵便はがき

102-00

東京都千代田区平河町2-4-14
平河町KSビル

株式会社 翔泳社

月刊プログラマーズ・ページ
広告資料部 行

2	受付No.	

マーズ・ページ広告資料請求カード

＊すでに会員番号をお持ちの方はご記入ください。

名	5	年齢	歳

綴じ込みの

創刊記念 HANDY 98 プレゼント

□10文系学生　□11その他(　　　　　　　)　ユーザー

創刊記念プレゼント記入欄	クイズ・ランダムスクリーン回答欄
希望商品番号	答

6	自宅住所	〒　　電話　（　　）
7	会社住所	〒　　電話　（　　）

資料送付希望先の番号に○印をつけてください。

オブジェクト指向の就職読本 第2回

WE WANT POWER!

▶会社が求めるスタッフ◀
公共機関の心臓を司る企業の場合

宮公庁、銀行、病院など、我々の生活と密着した公共機関でも、数多くのコンピュータがフル稼働している。これらはまさに「日常生活の心臓」だ。そして当然、心臓を動かす選りすぐりの技術者たちがいる。今回の特集は、我々の生活を支える企業やプログラマー達の実態を、現場サイドから探ってみた。

●どういう機構になっているのか

公共機関の業務を支える大型コンピュータ。これらのシステムを開発してるのは、公共機関自身ではなく、出入業者たちだ。

まず株式上場の大企業が公共機関からシステム開発を委託され、その大企業が子会社や中小企業に受けた仕事を流し、更にその細部が契約先の技術者に委託される。公共機関に王手をかけるために、たくさんの駒が、自分の持ち分を踏まえながら頑張っているということだ。どの駒に属するかによって、業務内容から意識まで大幅に違う。では現場の人達は、どんな意識を持っているのだろうか。

●責任が顔に出る大手の技術者

「ちょっと大袈裟かもしれないけど、"自分達が日本を背負って立っているんだ"って気構えが心のどっかにありますね。極端な話、我々の間違いがもとで、日本中の金融業務がパニックになりかねないわけですから。」とは、銀行の通貨処理システムの企画・開発を担当するスタッフ談。

彼の勤める会社は、日本の経済を取り仕切る大蔵省や日本銀行が取り引き先である。金融の自由化高度情報化社会の到来に合わせて、大蔵省や日銀のニーズは刻々と変化する。相手の要求を的確に呑みこむことが急務になる。その責任は重い。

また、同じく日銀や大蔵省を取引先に持つ、某通信機器メーカーのB氏はこう語る。「こういった仕事は、プロジェクトのスタンスが長いんです。1年2年は序の口で、5年とか10年とか。よく飽きないなあ、っていわれてるけど、やってる方は緊張とやり甲斐に生きる毎日ですよ。公共機関の要求を察知し、下請け会社を統率する。両方にアンテナはってるわけですから。」

では社員には何が求められるのか、と聞くと、

「社員は技術云々より、責任感と統率・管理能力が問われるんじゃないかと思います。特に下請けとの仕様に関する打ち合せは大変ですね。僕らが必要な情報を正確に相手に伝えられないと、膨大な数の人間が無駄な作業に時間とお金を費やしてしまう。結構しんどいですよ。でも、重要な責務を担っていることに誇りがある。この仕事、気に入ってます。」と笑顔のB氏だった。

●実践は任せろ!―中小企業の精鋭部隊

さて、こうして大企業が受注したシステム開発を、実際に手掛けている現場、中小企業側の実態はどうなのだろう。

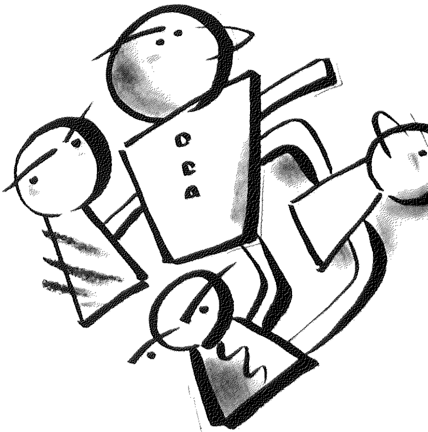
「請けた仕事を忠実にこなすのは、簡単なようで大変なんですよ。頭は使う、気は遣う。出向先で何をどうしたいか、どうしてほしいかなど、うまいことコミュニケーション取って相手のニーズを理解しないと、仕事はスムーズにいきません。」とはC氏談。

「僕の同期でいわばクビになったやつがいたんだけど、彼すごく頭はよかったんです。でもどちらかというと研究者肌で、内に閉じこもっちゃうタイプだったんですね。で、取引先とのやり取りが全然出来なくて、それイコール"仕事出来ない"って判断下されて終り。頭脳より対人処理能力が必要なんです。」

その人の持っている技術より、人間性が重要ということだ。

「でも技術より対人処理能力とはいっても、資格は持ってたほうが有利ですよ。誰だって資格ある人とない人じゃ、ある方が信頼できるでしょ?我々のような立場の企業は、有資格者を前面に出して仕事を請け負う傾向にあるから、いってみれば看板として、資格取っとくことをお勧めします。」

技術の最先端を追い求める人より、人と人との和、信頼を大切にしたいという人にはやり甲斐のある仕事に違いない。



●規模は小さくても、技術者魂は超一流

さらにその細部を請け負う小規模企業の様子はどうだろう。

「中には、取引先にペコペコしっぱなしで大変そうな会社もありますけど、僕らの会社は違いますね。小人数である分、一致団結して大手を見返してやろうという気概すらありますよ。それに、大きい企業と違って無駄な人材を抱えている余裕がない。一人一人が会社にとって重要で、欠かせない戦力なんです。ここにいる人間全員が、“自分が会社の顔だ”って思っているんじゃないでしょうか。」

独立心の旺盛さ、自分の腕一本で生きているという感覚、職人気質の技術屋が多いのが、小規模企業である。真ん中にいる中小企業より、個々の持つ技術の確かさが求められる割合は大きい。ある意味では、真の技術者とは、小規模企業にこそいる、と言えるかもしれない。

①(60%)	②(30%)	③(10%)

◆なぜ公共機関相手の企業を選んだのですか？

①安定しているから将来の不安がないし、自分の人生設計が立て易いから。
②大きなものを動かしてみたいと思ったから。
③世間的に聞こえがいいから。

①(40%)	②(40%)	③(10%)	④(10%)

◆どんな時に働き甲斐を感じてますか？

①与えられた仕事が満足いくように出来た瞬間。
②取引先の喜ぶ顔が見られたり、技術力を褒められた瞬間。
③街で自分の手掛けた製品をユーザーが使っているのを見た瞬間。
④仕事をやり終えたあとに待っている解放感。
休みには何をしようかなと考えている時。

①(50%)	②(20%)	③(10%)	④(10%)	⑤(10%)

◆これから何をしたいですか？

①人間関係や自分の才能を広げたい。　②出世。
③「暖簾分け」という形での独立。　④結婚してマイホームを持つ。
⑤将来の保証はなくなるけど、給料のいい「契約社員」になりたい

GET THE POLE POSITION

君のスターティンググリッド

公共機関の心臓を司るセクションに生きるには、どんな才能や人間性が要求されるのだろうか。

大企業・中小企業を問わず一番に言えるのは、『与えられた仕事を、確実にこなす責任感』だ。

技術的なことは心配ありません。研修制度がしっかり整ってますし、先輩に付いて実践していく中で、どんな人でも自然と身につきますから。」

と大判の企業が言う。実際、体育大学を出て、コンピュータのコの字も知らず入社した人が、2年たった今は第一線でバリバリ働いているそうだ。

次に重要なのは『コミュニケーション能力』。一つの大プロジェクトに携わる人の数は膨大だ。その橋渡しをするのが「言葉」である。橋が一つでも通じていなければ、そのプロジェクトは完璧に出来上がらないのだ。

更に、取材を通じて感じたのは、この仕事は『細かい押えがしっかりと出来る、抜けのない人向き』ということである。技術的なひらめきは素晴らしいが、抜けなく仕事をこなすのは少々苦手、という人にはあまり向かないかもしれない。いわば“技術屋界の公務員”になりたい人には、仕事面でも環境面でももってこいの職種である。

事情通の眼

現場の声

これがあこがれの「画期的企業」だ!!

就職を考えたとき、企業選択条件の筆頭に挙げられるのが、会社の業績と将来性である。しかも新たな飛躍を目指す諸君なら、脚光を浴びる「トップシェア企業」なんて憧れてしまうだろう。マーケットシェア(市場専有率)をめぐって、企業間では熾烈な争いがくりひろげられている。生き残るのも至難の技だ。そんな中、確実にトップクラスを維持し続けているソフト開発会社を覗いてみようと、編集部の意見がまとまった。

〈近未来空間を観た！〉

秋晴れの午後、我々はA社の中央研究所を訪ねた。〝総工費数十億〟と聞いていたからある程度想像していたが、見た瞬間タメ息が出た。広々とした庭園、200台は入ろうかという地下駐車場、柱のない開放的な空間、総木造りの社員食堂。大型プロジェクターなど最新鋭の機器を備えた多目的イベントホール。出来すぎている。

驚く我々を迎えてくれたのが、総務課長のN氏であった。

〈A社の飛翔過程〉

14年前、理想に燃えた5人の技術者があった。

彼らが手がけたのは「CADシステム開発」だった。(このシステム開発は当る！)彼らは確信していたのである。

1年半後。5人は独自の設計思想に基づいたシステムを生み出し注目を集め、会社は目標以上の急成長を遂げる。設立4年目にして国産初のシステムを開発し、売り上げは3億円。従業員は設立当初の64倍に膨れあがっている。

「これからも伸びるっていう確実な手応え感じます。」

そう言って総務課長N氏は笑った。

〈トップシェア獲得の秘密〉

何が会社をここまでおしあげたのだろうか。CADシステムの開発が、ニーズに合致していたともいえる。しかし、真に企業を大きくした要因は、開発したシステムではない、とN氏は語る。

「人。全ては人ですよ。」

そうN氏は言い切った。

A社では、その〝人〟が独創性と開拓者精神を存分に発揮できる自由な気風を保ち続けることこそ、会社の務めと考えている。前述の至れり尽くせりの環境の一つとっても、その親心がさりげなく伝わってくる。日々困難なテーマに立ち向かっている社員にとって、深呼吸できるくつろぎのスペースは何よりのプレゼントだろう。

「うちの会社の辞書はね、〝ジンザイ〟のザイは、財産の〝財〟になってるんですよ。」

N氏のジョークに、大きく頷いた我々だった。

〈新卒？中退？そんな区別は必要ない!〉

「それなら中途採用者も新卒同様、伸び伸びと働けますね。」

こちらの質問にN氏の答えがこれであった。

「オリジナリティと意欲と誇りのある人。採用の条件ってそれだけですよ。」

実際、すれちがう社員は皆はつらつとしている。

「また、我が社では徹底的なディスカッションを実行しています。ディスカッションの際は、例え部下だって自分の意見が正しいと思ったら遠慮しませんよ。上から降りてくる命令を下が従うだけでは、せっかくのオリジナリティも死んでしまいますから。」

「トップダウン」なんて言葉はハナっからない。「ボトムアップ」も太古の用語だ。上下関係なく、不満から改革案まで、本音でぶつけ合う。分かり合い、認め合った仲間は結束が固い。腹に溜めず言葉にでき、それが生かされるのだから、次々と前向きな意見が飛び出てくる。

〈続・どうやってユートピア企業を見極めるか〉

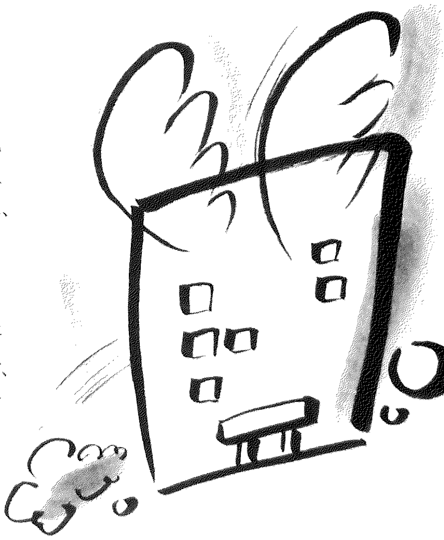
まさにこの会社は「理想」だ。

理想が「現実」に存在している。独創的で、活気があって、将来性絶対ありで、自らの技術を存分に発揮し研いてゆける。そんな〝ユートピア企業〟を何とか情報集めの段階で見極めたいものだ。

『シェアの急速な拡大を果たした企業』

独自の力でマーケットを発見し、自分たちの力でマーケットチャンスを手中にし、堂々と確固たる地位を確保した企業がそれと言える。会社四季報や求人情報誌で、売り上げの伸び率、製品開発の動向、そのマーケット自体の将来性、シェアを確実に維持・拡大しているかどうかなどをチェックするとよい。社員をどれほど大切にしているかは、施設や複利厚生面がバロメーターとなるだろう。そしていかに自由闊達な社風であるかは、社員を一目見れば分かる。A社の面々のように、輝く目をして弾むように歩いていたりしたら、大当りである。

AUTHORITY'S EYE



【ここだけの話】Q&A
会社選びのテクニック

第二の人生を考えるときは、誰しも慎重になる。"今度こそは判断を誤るまい" と思う。その「判断」を的確にするにはどうしたらいいか。会社選びに関するQ&A。

Q：「ああ、もう辞めてしまいたい！」っていう気持ちが強くなる一方なんですよ。でも次にどうしたらいいのかを考えると「待てよ。俺はいったい何がやりたいんだ?」ってなっちゃうんです。なんとか自分の方向性を早く見つけたい。それが見えてくる方法って、なにかないですか?

A：一番分かっているのが「自分」。自分が本当は何をしたいのかをちゃんと知っている人って、実は案外少ないんだよね。でも、みんなが出来ていないから自分もできなくていいってわけじゃない。答は見つけたい。じゃあどうやって答を見つけるか。

一つは「不満を分析してみること」。つまり自分のグチを聞いてあげるわけ。今の会社、何が不満なのかな。仕事の内容? 人間関係? それとも自分の能力をちゃんと給料で評価してもらえない事かな? それが分かったら、そうじゃないとこ、その不満を解消してくれるとこに行けばいいってことになるわけでしょ。

二つ目は「自分の興味・関心の方向を分析してみること」。例えば求人雑誌みて自分の気になる箇所はどこか。働きたいって少しでも思える会社はどんな会社か。その傾向が摑めれば、次の会社の業務内容が絞れるよね。

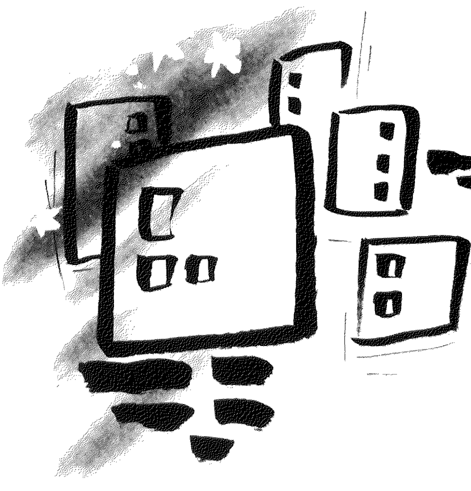
三つ目は「自分の適正や得意分野を確認してみること」。何をするのが好きなのか。どんなことに自信があるか。要は積極的に、楽しく仕事しなきゃ意味がないわけだから、肝心なのはそこなんだよね。"職探し" って考えるから、頭が堅くなっちゃっう。気楽な気分で探してみよう。真面目になりすぎるからかえってうまくいかない人って、結構世の中には多いと思うよ。

焦らないで、自分に対して敏感になることから始めてみよう。

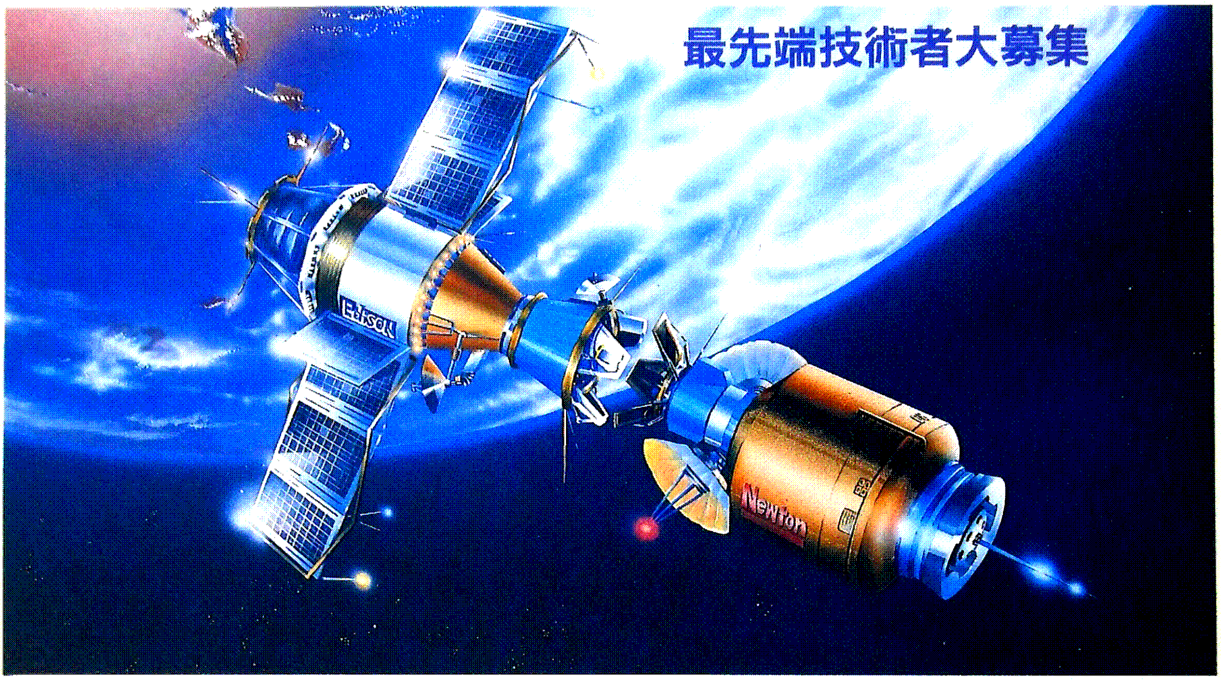
Q：分厚い求人情報誌手に取るたんびに、ため息が出ちゃう。なぜって、多すぎるんだもの、おなじような条件の会社が。特に技術系ってそうなんだよね。どこをどう選んだらいいんだか、目移りばっかりしている俺。変な悩みだけど、正直困ってるんだ。

A：いいんじゃないの、そんなにたくさんの中から思うままに選べるんだから。学生の時みたいに、ま、気持ち分かるけどね。

さてさて、第1条件である会社の規模・業種・業績、それから業務内容や資格なんかがほとんど変わらない会社がたくさんあったら、次に見るのは何かってことだよね。え? やっぱ金だって? それも言える。でも実際は、給料の相場って割と決まってるんだよね。で、何を判断基準にしたらいいかっていうと、「勤務地」。通勤時間の多い少ないは、仕事に微妙に影響してくる。行って帰ってくるだけで疲れちゃったら、何もならないでしょう。それでも近くに寮や社宅があれば問題ない。ということは「福利厚生」にも注目だね。逆に、「経験」とか「研修制度やその期間」に関しては、あまり神経質にならなくていいと思う。会社としてはきっちり仕事をしてもらわなきゃ困るわけだから、戦力になるまでしっかり育ててくれるよ。



インテリジェントな未来のために。



未経験者1月生募集(定員30名)

(中野研修センター内で3ヵ月間の未経験者教育を行います)

■未経験者大募集■元年・2年既卒者(大・短・専)同時募集

教育部の専任講師による3ヵ月間の集合教育を行います。まずコンピュータの構造を知るための基礎教育に始まり、1人1台の実習機を使っての教育を実践しています。着実に実力が身につくカリキュラム化した学習内容がありますので、安心して望むことができます。

本社(新築ビル)

●NEWSONは、新時代創造会社です。

NEWSONは今年創立17年を迎えるシステムハウス。新時代を築くシステムインテグレータとして様々な分野で活躍を続けています。メインフィールドである通信・制御部門では衛星通信やISDNをはじめとする各種通信システムを手掛け、金融情報部門では株式・債券・通貨などの先端情報システムを。そして大型汎用系情報処理システムやハードウェア、最近ではパソコンを使った教育システムを開発するなど、着実に確実に、その活動範囲を広げています。

●NEWSONは、技術者のための会社です。

NEWSONは一括受託型。だから当然、出向や派遣はありません。技術者たちは、システム調査、分析、基本設計からシステム運用に至るまで、一連の作業を手掛けることができます。また、平均年齢が若く、何でも言える自由な雰囲気もNEWSONの大きな特徴のひとつです。NEWSONをはじめて訪れる方は、その環境の良さにきっと驚かれることでしょう。まずは遊びに来て下さい。

世界へ宇宙へ、ニュートンとエジソンの英知を結集

NEWSON

ニューソン・コンピュータ株式会社

■会社概要

設　　立／昭和49年2月
資 本 金／3000万円
売 上 高／26億円(平成2年7月)
従業員数／350名
事 業 所／本社(渋谷)、中野、横浜、大阪、神戸
取 引 先／東芝、日本電気、沖電気工業、松下通信工業、三菱電機、日本DEC、日本IBM、他、銀行・証券
関連会社／ニューソン・ソフトウェア㈱
事業内容／通信・制御、金融情報系、各種事務処理およびハードウェア開発

■募集事項

職　　種／ソフトウェア技術者、ハードウェア技術者、営業及び技術営業、上級管理者
資　　格／高卒以上、45歳位迄の男女
給　　与／㊍㊊17万円～50万円
27歳経験7年通信系中級SEの場合
㊍㊊33万円家族手当(既婚、子1人)、住宅手当、情報処理資格手当(1種)、職務手当含む。年収558万円(元年実績賞与6ヶ月含む)
※賃貸住宅の方別途1万円支給
契約社員5年以上、月35万円～70万円
待　　遇／昇給年1回、賞与年2回、情報処理資格手当(2種5000円、1種10000円、特種30000円)、精勤手当、住宅手当、帰省手当、家族手当(配偶者2万円、子供2人迄1人につき1万円)、㊂全支給、各種社会保険完備、退職金、財形援助、社内研修、社員旅行、ゴルフ、テニス、ボウリング、野球等
休日休暇／完全週休2日制、祝日、夏季(12日)、年末年始(9日)、有給、年間休日121日
勤 務 地／本社(渋谷)、ほか各事務所
勤務時間／9:00～17:30
応　　募／履歴書(写真貼付)、業務経歴書(経験者のみ)を郵送又は持参

■連絡先／〒150 東京都渋谷区東1-32-12 渋谷プロパティーウエストビル6F
☎03(3407)9481
総務部／内野、関山

新たなる未知の果実を生み出すために
先端的頭脳の種子を求めています

たとえば、人間が初めて絵を描いたとき、小さな平面に表現される世界にどれほどの感動を覚えたことでしょう。初めて道具を使ったとき、自分の身体以外の可能性にどれほど驚いたことでしょう。未知のものに形を与える。私たちはその喜びを胸に歩んで来ました。考えてみると、私たちの周りはさまざまなイメージで満ちあふれています。多様化する価値観、目まぐるしく移り変わる流行。その中に秘められた可能性をひとつの形としてつくりあげること、それは現代に生きる私たちにとっても変わることのない大きな喜びです。

21世紀を目前にして、私たちは、大きな転換期ともいうべき時代を迎えています。
国際的な社会情勢の急激な変化、為替の変動、貿易摩擦、エネルギー問題など、経済環境は年々予測が難しくなっています。また、エレクトロニクス技術の発達によって、社会のすみずみまでマイクロコンピュータ制御がいきわたり、あらゆる分野で自動化が進んでいます。このような時代にあって、ものの価値、ものをつくることの意味も大きく変わろうとしています。

人材募集
新卒者・経験者募集

ものを「つくる」時代から、新たな価値を「創造」する時代へ。ものを「つくる」ためにテクノロジーの粋を集めてそそぎ込むだけでは、もはや十分とは言えません。激動する時代の流れを素早く読みとって、新たな価値、新たな意味を探し出し、誰よりも早く未知の果実をひとつの形に実らせる力が必要です。

CIM（Computer Integrated Manufacturing）も、これからの経営戦略の成否を決定する重要な鍵となります。最新のテクノロジーと情報化という概念の構築、そして人間の創造力を統合してはじめて、このマネージメントは実現します。

このように情報化の中で生まれる新たなる未知の果実。私達は常に進化し続ける状況に対応すべく、柔軟な発想と絶え間ない最先端技術とを追求し続けています。そしてそれゆえに、TAMの企業活動の基本は、それぞれの個人がいかに快適に、いかに創造的に仕事に向かうことができるか、ということなのです。この環境の中で育った種子は、新たな果実となり、既存の価値や関係を革新していくことでしょう。未来のフロダクツ、未来のマネジメントを思考するすべての人へ、私達は新たな価値創造を提言します。

TAM

会社概要

社名	タム株式会社
創業	昭和49年2月
設立	昭和58年7月
資本金	5,000万円
代表取締役社長	市原静司
事業内容	CAD/CAM, CIMのシステムインテクレーション，及ひコンサルテーション
年商	27億円(平成2年3月実績)
社員数	90名
主要取引先	トヨタ自動車，本田技研工業，日産自動車，松下電器産業，松下電工，キヤノン，ニコン，オリンハス光学工業，三菱重工，東芝，アルフス電気，トヒー工業，日本IBM，メティエマネシメント社ほか
主要取引銀行	第一勧業銀行，太陽神戸三井銀行，東京銀行，埼玉銀行

待遇

初任給	大卒(22才) 182,000円(平成2年4月) 短大，専門校卒(20才) 162,000円(平成2年4月)
昇給	年1回(4月) 約5%
賞与	年2回(6月と12月) 入社年度 3.5ヶ月(平成2年度実績) 入社1年以上 8ヶ月(平成2年度実績)
手当	時間外手当，住宅手当，通勤手当(全額)
勤務時間	9:00〜17:30(たたし,開発部はフレックスタイム制度)
休日休暇	完全週休二日制(土，日曜日)，夏季，年末年始，特別有給休暇

福利厚生

1 健康保険，厚生年金保険，雇用保険，労災保険加入

2 退職金制度，及び退職年金制度

3 施設　リソートウィラ法人会員(宿泊施設,日本全国及ひ海外あり)
スポーツクラフ法人会員(トゥ・スポーツフラサほか)
テニスクラフ法人会員

4 社員親睦会　(タム友の会)
3年に1回は海外旅行(シンカホール，香港，クァムの実績あり)，その他年1回国内旅行，コルフコンペ，野球大会，ボーリンク大会，テニス，スキー等盛沢山

5 独身寮完備

6 持株制度

事業所

■本社　〒160 東京都新宿区西新宿4丁目15番7号 後楽園新宿ヒル
Phone 03-3299-8811(大代表)

■大阪営業所　〒540 大阪市中央区城見1-2-27 クリスタルタワー
Phone 06-949-5060代

■名古屋営業所　〒465 名古屋市名東区上社2-210 北村第二ヒル
Phone 052-775-5721代

応募連絡先

TEL 03(3299)8811　総務部　吉村，高島

あなたの情報アンテナを
本づくりへ

編集スタッフ募集

募集要項

職種／雑誌・書籍・マニュアルの編集　**資格**／PCのプログラミング言語・OSについての基礎知識を有し，文章力に自信のある編集経験者あるいは本づくりに関心のあるソフトウェア技術者．　**給与**／平成２年大卒初任給200,000円．経験・年齢・能力を考慮の上，当社規定により優遇．中途採用のハンディは，一切ありません．　**待遇**／昇給年1回，賞与年2回．住宅手当・家族手当・残業手当有り．各種社会保険完備．交通費全額支給．　**休日**／週休2日制．　**休暇**／春・夏・冬それぞれ7～10日間の連休有り．　**勤務**／10:00～18:00
応募／履歴書をご郵送下さい．追って詳細をご連絡致します．

会社概要

事業内容●コンピュータ・エレクトロニクス・情報処理関係の書籍・雑誌出版　●大手メーカーのソフトウェアマニュアルの企画・編集制作　●CIをはじめとする各種デザイン事業　**設立**／昭和60年12月19日　**資本金**／2,000万円
売上高／13億5,600万円（平成元年度）

SE

株式会社 翔泳社　〒102 東京都千代田区平河町2-4-14 平河町KSビル4F・5F TEL.03-3263-0447 担当：人事係

MS-Windows 3.0関数一覧　（③＝Ver.3.0から加わった関数）

■Window Manager Interface Functions

●Message functions

CallWindowProc	メッセージの情報を指定した関数へ渡す。
DispatchMessage	指定したウィンドウ関数へメッセージを渡す。
GetMessage	指定した範囲からメッセージを取得する。
GetMessagePos	最後にメッセージを取得したときのマウスの座標を返す。
GetMessageTime	最後にメッセージを取得したときの時刻を返す。
InSendMessage③	現在のウィンドウ関数が、SendMessage関数で受け取ったメッセージを処理しているかを調べる。
PeekMessage	アプリケーションのキュー内のメッセージを調べる。
PostAppMessage	アプリケーションへメッセージを送る。
PostMessage	アプリケーションキューにメッセージを入れる。
PostQuitMessage	アプリケーションへWM_QUITメッセージを送る。
ReplyMessage	メッセージに応答する。
SendMessage	ウィンドウにメッセージを送る。
SetMessageQueue	異なるサイズの新しいメッセージキューを作成する。
TranslateAccelerator	メニューコマンドのためにアクセレータキーを処理する。
TranslateMDISysAccel③	マルチドキュメントインターフェース（MDI）の子ウィンドウコマンドアクセレータを処理する。
TranslateMessage	仮想メッセージを文字メッセージに変換する。
WaitMessage	他のアプリケーションに制御を渡す。
WinMain	Windowsアプリケーションを実行するために、エントリポイントを提供する。

●Window-creation functions

AdjustWindowRect③	あたえられたクライアント領域と一致するウィンドウサイズを計算する。
AdjustWindowRectEx③	あたえられたクライアント領域と一致するように、拡張スタイルのウィンドウサイズを計算する。
CreateWindow	オーバーラップウィンドウ、ポップアップウィンドウ、子ウィンドウを作成する。
CreateWindowEx③	拡張スタイルのオーバーラップウィンドウ、ポップアップウィンドウ、子ウィンドウを作成する。
DefDlgProc③	アプリケーションが処理しないダイアログボックスメッセージのデフォルトの処理をする。
DefFrameProc③	アプリケーションが処理しないMDIのフレームウィンドウメッセージのデフォルトの処理をする。
DefMDIChildProc③	アプリケーションが処理しないMDIの子ウィンドウメッセージのデフォルトの処理をする。
DefWindowProc	アプリケーションが処理しないデフォルトのメッセージ処理を用意する。
DestroyWindow	WM_DESTROYメッセージを送出する。
GetClassInfo③	指定したクラスについての情報を取得する。
GetClassLong	ウィンドウクラス情報をWNDCLASS構造体から取得する。
GetClassName	ウィンドウクラス名を取得する。
GetClassWord	ウィンドウクラス情報をWNDCLASS構造体から取得する。
GetLastActivePopup③	別ウィンドウが所有するポップアップウィンドウのうちで、最後までアクティブだったものを取得する。
GetWindowLong	ウィンドウについての情報を取得する。
GetWindowWord	ウィンドウについての情報を取得する。
RegisterClass	ウィンドウクラスを登録する。
SetClassLong	WNDCLASS構造体の情報を置き換える。
SetClassWord	WNDCLASS構造体の情報を置き換える。
SetWindowLong	ウィンドウの属性を変更する。
SetWindowWord	ウィンドウの属性を変更する。
UnregisterClass③	ウィンドウクラステーブルからウィンドウクラスを削除する。

●Display and movement functions

ArrangeIconicWindows③	最小化、またはアイコン表示された子ウィンドウを並べる。
BeginDeferWindowPos③	DeferWindowPos関数が使うメモリを初期化する。
BringWindowToTop③	指定したウィンドウをオーバーラップしたウィンドウの中で最上位のウィンドウにする。
CloseWindow	指定したウィンドウを閉じる、または、アイコン化する。
DeferWindowPos③	EndDeferWindowPos関数で移動またはサイズ変更されたウィンドウの位置を記録する。
EndDeferWindowPos③	DeferWindowPos関数が記録している情報に基づいて、同時にいくつかのウィンドウの移動またはサイズを変更する。
GetClientRect	ウィンドウのクライアント領域の座標をコピーする。
GetWindowRect	ウィンドウ全体をコピーする。
GetWindowText	ウィンドウのキャプションをコピーする。
GetWindowTextLength	あたえられたウィンドウのキャプションの長さを返す。
IsIconic	ウィンドウがウィンドウとして表示されているか、アイコンとして表示しているかを指定する。
IsWindowVisible	あたえられたウィンドウが可視かどうかを調べる。
IsZoomed	そのウィンドウがフルスクリーン表示されているかどうかを調べる。
MoveWindow	そのウィンドウのサイズと位置を返す。
OpenIcon	指定したウィンドウをオープンする。
SetWindowPos	子ウィンドウかポップアップのサイズ、位置、順番を変更する。
SetwindowText	ウィンドウのキャプションを設定する。
ShowOwnedPopups	全てのウィンドウを見せるか隠す。
ShowWindow	あたえられたウィンドウを表示するか削除する。

●Input functions

EnableWindow	アプリケーションを通じて、マウスとキーボードの入力を有効にしたり無効にしたりする。
GetActiveWindow	アクティブウィンドウへのハンドルを返す。
GetCapture	マウス入力をしているウィンドウのハンドルを返す。
GetCurrentTime③	現在のWindows時刻を調べる。
GetDoubleClickTime③	マウスの現在のダブルクリック時間を取得する。
GetFocus	現在、入力フォーカスをもつウィンドウへのハンドルを取得する。
GetTickCount	システムがブートされてからタイマーがカウントした数を返す。
IsWindowEnabled③	ウィンドウがマウス入力とキーボード入力を許可しているかを調べる。
KillTimer	指定したタイマーイベントを終了する。
ReleaseCapture	マウスの入力を解放し、通常の入力処理に戻す。
SetActiveWindow	ウィンドウをアクティブにする。
SetCapture	マウスの入力を指定したウィンドウに送るようにする。
SetDoubleClickTime③	マウスのダブルクリック時間を設定する。
SetFocus	入力フォーカスを指定したウィンドウに割り当てる。
SetSysModalWindow	指定したウィンドウをシステムモード付きのウィンドウにする。
SetTimer	システムタイマーのイベントを作成する。
SwapMouseButton③	左右のマウスボタンの機能を交換する。

●Hardware functions

EnableHardwareInput	アプリケーションを通じて、マウスとキーボードの入力を有効にしたり無効にしたりする。
GetAsyncKeyState	キーの状態の割り込みレベルの情報を返す。
GetInputState	マウスかキーボードの入力があれば、TRUEを返す。
GetKBCodePage③	OEM/ANSIのどちらのテーブルがロードされているかを調べる。
GetKeyboardState	キーボードのキーの状態を含んだ配列をコピーする。
GetKeyNameText③	キーボードドライバが管理しているリストから、キーの名前を含む文字列を取得する。
GetKeyState	仮想キーの状態を取得する。
MapVirtualKey③	仮想キーコード、またはスキャンコードを受け取り、対応するスキャンコード、仮想キーコードまたはASCII値を返す。
OemKeyScan③	0から0xFFのOEM ASCIIコードをOEMスキャンコードとシフト状態にマップする。
SetKeyboardState	配列の値を変えて、キーボードのキーの状態を設定する。
VkKeyScan③	ANSI文字を、現在のキーボードの仮想キーコードとシフト状態に変換する。

●Painting functions

BeginPaint	ウィンドウが描画する準備をする。
DrawFocusRect③	フォーカスを表すスタイルの矩形を描画する。
DrawIcon	アイコンを描画する。
DrawText	指定した文字列を描画する。
EndPaint	ウィンドウの再描画の終りをマークする。
ExcludeUpdateRgn	ウィンドウ内の不正な領域に描かないようにする。
FillRect	指定されたブラシを使ってあたえられた矩形を塗りつぶす。
FrameRect	あたえられた矩形の周りに境界を描く。
GetDC	クライアント領域のためにディスプレイコンテキストを取得する。
GetUpdateRect	ウィンドウの更新領域を境界の矩形の大きさにコピーする。
GetUpdateRgn	ウィンドウの更新領域をコピーする。
GetWindowDC	ウィンドウ全体のためにディスプレイコンテキストを取得する。
GrayString	淡色表示テキストを使って文字列を書く。
InvalidateRect	再描画のために矩形にマークを付ける。
InvalidateRgn	再描画のために領域にマークを付ける。
InvertRect	指定した矩形のビットを反転表示する。
ReleaseDC	ディスプレイコンテキストを解放する。
UpdateWindow	ウィンドウの一部を書き直す必要が生じたとき、アプリケーションに通知する。
ValidateRect	再描画の指定から矩形を解放する。
ValidateRgn	再描画の指定から領域を解放する。

●Dialog-box functions

CheckDlgButton	チェックを付けるか取り除く、あるいは3つの状態をもつボタンの状態を変更する。
CheckRadioButton	指定したボタンチェックし、他の全部のチェックを取り除く。
CreateDialog	モードなしダイアログボックスを作成する。
CreateDialogIndirect	テンプレートからモードなしダイアログボックスを作成する。
CreateDialogIndirectParam③	テンプレートからモードなしダイアログボックスを作成しデータを渡す。
CreateDialogParam③	モードなしダイアログボックスを作成しデータを渡す。
DefDlgProc③	プライベートウィンドウクラスをもつダイアログボックスが処理しないWindowsメッセージのデフォルトの処理をする。
DialogBox	モード付きダイアログボックスを作成する。
DialogBoxIndirect	テンプレートからモード付きダイアログボックスを作成する。
DialogBoxIndirectParam③	テンプレートからモード付きダイアログボックスを作成しデータを渡す。
DialogBoxParam③	モード付きダイアログボックスを作成しデータを渡す。
DlgDirList	パスと一致するファイル名をリストボックスで一覧表示する。
DlgDirListComboBox③	コンボボックスにパスと一致するファイル名を一覧表示する。
DlgDirSelect	リストボックスで指定されているものを文字列にコピーする。
DlgDirSelectComboBox③	コンボボックスで指定されているものを文字列にコピーする。
EndDialog	リソースを解放し、モード付きダイアログボックスに関連したウィンドウを消去する。
GetDialogBaseUnits③	ダイアログボックス作成時に、Windowsが使う基本ダイアログ単位を取得する。
GetDlgCtrlID③	コントロールウィンドウのIDを返す。
GetDlgItem	あたえられたダイアログボックスからダイアログ項目のハンドルを取得する。
GetDlgItemInt	ある項目のテキストを整数値に置き換える。
GetDlgItemText③	ある項目のコントロールテキストを文字列バッファにコピーする。
GetNextDlgGroupItem	そのグループ内のつぎの項目のウィンドウハンドルを返す。
GetNextDlgTabItem	そのグループ内の、つぎか前の項目のウィンドウハンドルを返す。
IsDialogMessage	メッセージがそのダイアログボックスのためのものかを調べる。
IsDlgButtonChecked	ボタンがチェックされているかを調べる。
MapDialogRect	ダイアログボックスの座標をクライアント座標に変換する。
SendDlgItemMessage	ダイアログボックス内の項目にメッセージを送る。
SetDlgItemInt	項目のテキストに整数を表現している文字列に設定する。
SetDlgItemText	キャプションや項目のテキストを文字列に設定する。

●Scrolling functions

GetScrollPos	スクロールバーボックスの現在の位置を取得する。
GetScrollRange	指定されたスクロールのためにあたえられたスクロールバーの最小と最大の位置をコピーする。
ScrollDC	画面または矩形を水平および垂直にスクロールさせる。
ScrollWindow	クライアント領域の内容を移動させる。
SetScrollPos	スクロールバーボックスを設定する。
SetScrollRange	スクロールバーの最小と最大の位置を設定する。
ShowScrollBar	スクロールバーの表示またはコントロールを表示するか削除する。

●Menu functions

AppendMenu③	メニュー項目をメニューに追加する。
CheckMenuItem	ポップアップメニューの項目のとなりに、チェックマークを付けたり取り除いたりする。
CreateMenu	空のメニューを作成する。
CreatePopupMenu③	空のポップアップメニューを作成する。
DeleteMenu③	メニュー項目を削除し、付随するポップアップメニューも削除する。
DestroyMenu	指定したメニューを閉じる。
DrawMenuBar	メニューバーを描き直す。
EnableMenuItem	メニュー項目を有効、無効にしたり、淡色表示したりする。
GetMenu	指定したウィンドウのメニューへのハンドルを取得する。
GetMenuCheckMarkDimensions③	ディフォルトのチェックマークのビットマップの寸法を取得する。
GetMenuItemCount	メニュー内の項目のカウントを返す。
GetMenuItemID	メニュー項目のIDを返す。
GetMenuState	メニュー項目の状態を取得する。
GetMenuString	メニューラベルを文字列にコピーする。
GetSubMenu	ポップアップメニューのメニューハンドルを取得する。
GetSystemMenu	コピーと変更のために、システムメニューにアクセスする。
HiliteMenuItem	トップレベルのメニューを見えるようにしたり、見えないようにしたりする。
InsertMenu③	メニュー項目をメニューに挿入する。
LoadMenuIndirect	メニューのリソースをロードする。
ModifyMenu③	メニュー項目を変更する。
RemoveMenu③	メニューから項目を削除する。メニューは消去しない。
SetMenu	新しいウィンドウメニューを指定する。
SetMenuItemBitmaps③	メニュー項目にチェックマークを付けたり、はずしたりするときの表示のためにメニュー項目とビットマップを対応させる。
TrackPopupMenu③	指定した画面上に、ポップアップメニューを表示し、メニューへのユーザー操作を追跡する。

●Information functions

AnyPopup	ポップアップウィンドウが見えるかを示す。
ChildWindowFromPoint	どの子ウィンドウが指定した点を含んでいるかを調べる。
EnumChildWindows	指定した親ウィンドウに属する子ウィンドウを列挙する。
EnumTaskWindows	タスクに関連するすべてのウィンドウを列挙する。
EnumWindows	ディスプレイ上のウィンドウを列挙する。
FindWindow	あたえられたクラスとキャプションのウィンドウのハンドルを返す。
GetNextWindow	つぎか前のウィンドウにハンドルを返す。
GetParent	指定したウィンドウの親ウィンドウのハンドルを取得する。
GetTopWindow	トップレベルの子ウィンドウのハンドルを返す。
GetWindow	ウィンドウマネージャーのリストからハンドルを返す。
GetWindowTask	ウィンドウに関連するタスクのハンドルを返す。
IsChild③	ウィンドウが指定ウィンドウの子孫であるかを判断する。
IsWindow	ウィンドウが有効か、または存在するかを調べる。
SetParent	子ウィンドウの親を変更する。
WindowFromPoint	指定した点を含んでいるウィンドウを識別する。

●System functions

GetCurrentTime	システムがブートしてからの時間を返す。
GetSysColor	システムカラーを取得する。
GetSystemMetrics	システムメトリックに関する情報を取得する。
SetSysColors	1つ以上のシステムカラーを変更する。

●Clipboard functions

ChageClipboardChain③	クリップボードチェインからウィンドウを削除する。
CloseClipboard	クリップボードをクローズする。
EmptyClipboard	クリップボードを空にし、クリップボードの所有権を変える。
EnumClipboardFormats	利用可能なクリップボードのフォーマットを列挙する。
GetClipboardData	クリップボードからデータを取得する。
GetClipboardFormatName	クリップボードのフォーマットを取得する。
GetClipboardOwner	現在のクリップボードの所有権に関連するウィンドウのハンドルを取得する。
GetClipboardViewer	クリップボードビュアチェインの中で最初のウィンドウのハンドルを取得する。
GetPriorityClipboardFormat③	優先順位が付けられたフォーマットリストの先頭のフォーマットにより、クリップボードからデータを取得する。
IsClipboardFormatAvailable	あたえられたフォーマットが使えるなら、TRUEを返す。
OpenClipboard	クリップボードをオープンする。
RegisterClipboardFormat③	新しいクリップボードフォーマットを登録する。
SetClipboardData	データのハンドルをコピーする。
SetClipboardViewer	クリップボードビュアチェインにハンドルを追加する。

●Error functions

FlashWindow	交互にアクティブ状態、アクティブでない状態にすることによってウィンドウを点滅させる。
MessageBeep	システムのスピーカからビープ音を発生させる。
MessageBox	あたえられたテキストとキャプションのウィンドウを作成する。

●Caret functions

CreateCaret	キャレットを作成する。
DestroyCaret	現在のキャレットを消去する。
GetCaretBlinkTime	キャレットの点滅周期を返す。
GetCaretPos	現在のキャレット位置を返す。
HideCaret	あたえられたウィンドウからキャレットを削除する。
SetCaretBlinkTime	キャレットの点滅周期を設定する。
SetCaretPos	指定したキャレットを動かす。
ShowCaret	新しく作成されたキャレットを表示したり、隠れているキャレットを再表示する。

●Cursor functions

ClipCursor	マウスカーソルが動ける範囲をあたえられた矩形内に制限する。
CreateCursor③	2個のビットマスクでカーソルを作成する。
DestroyCursor③	CreateCursor関数で作成したカーソルを破棄する。
GetCursorPos	マウスカーソルのスクリーン座標を設定する。
LoadCursor	リソースファイルからカーソルをロードする。
SetCursor	カーソルの形を設定する。
SetCursorPos	マウスカーソルの位置を設定する。
ShowCursor	カーソルディスプレイカウントを増加、または減少させる。

●Hook functions

CallMsgFilter③	現在のメッセージフィルタ関数にメッセージやその他のデータを渡す。
DefHookProc	フィルタ関数チェインの中のつぎのフィルタ関数を呼び出す。
SetWindowsHook	システムとアプリケーション、あるいは一方のフィルタ関数をインストールする。
UnhookWindowsHook	Windowsのフィルタ関数をフィルタ関数チェインから削除する。

●Property functions

EnumProps	ウィンドウのプロパティを列挙関数に渡す。
GetProp	ウィンドウプロパティリストから文字列に結び付けられたハンドルを取得する。
RemoveProp	プロパティリストから文字列を削除する。
SetProp	文字列とデータハンドルをウィンドウプロパティリストにコピーする。

●Rectangle functions

CopyRect	薬手にある矩形のコピーを作成する。
EqualRect	2つの矩形が同じであるかを調べる。
InflateRect	指定した矩形を拡大または縮小する。
IntersectRect	2つの矩形の重なる部分を探す。
OffsetRect	あたえられた矩形を動かす。
PtInRect	あたえられた矩形に指定した点があるかを知らせる。
SetRectEmpty	矩形の内容を空に設定する。
UnionRect	2つの矩形を結合して設定する。

■Graphics Device Interface Functions

●Device-context functions

CreateCompatibleDC	メモリデバイスコンテキストを作成する。
CreateDC	デバイスコンテキストを作成する。
CreateIC	インフォメーションコンテキストを作成する。
DeleteDC	デバイスコンテキストを削除する。
GetDCOrg③	デバイスコンテキストの原点を取得する。
RestoreDC	デバイスコンテキストを再設定する。
SaveDC	デバイスコンテキストの現在の状態をセーブする。

●Drawing-tool functions

CreateBrushIndirect③	論理ブラシを作成する。
CreateDIBPatternBrush③	デバイスに依存しないビットマップによって定義されたパターンの論理ブラシを作成する。
CreateHatchBrush	ハッチパターンをもつ論理ブラシを作成する。
CreatePatternBrush	指定したパターンをもつ論理ブラシを作成する。
CreatePen	論理ペンを作成する。
CreatePenIndirect	論理ペンを作成する。
CreateSolidBrush	論理ブラシを作成する。
DeleteObject	論理ペン、ブラシ、フォント、ビットマップ、領域を削除する。
EnumObjects	利用可能なペンやブラシを列挙する。
GetBrushOrg③	デバイスコンテキストの現在のブラシの原点を取得する。
GetObject	オブジェクトを定義する論理データのバイトをコピーする。
GetStockObject③	定義済みのストックペン、ストックブラシ、ストックフォント、カラーパレットへのハンドルを取得する。
SelectObject	あるオブジェクトを現在のオブジェクトに変換する。
SetBrushOrg③	デバイスコンテキストで選択されたすべてのブラシの原点を設定する。
UnrealizeObject	GDIに与えられたブラシの原点を再設定する。

●Color-palette functions

AnimatePalette③	論理パレット内のエントリを置き換える。Windowsは新しいエントリをすぐにシステムパレットに対応させる。
CreatePalette③	論理パレットを作成する。
GetNearestPaletteIndex③	指定したRGB値に最も近い論理パレットエントリを取得する。
GetPaletteEntries③	論理パレットエントリを取得する。
GetSystemPaletteEntries③	システムパレットからパレットエントリの範囲を取得する。
GetSystemPaletteUse③	アプリケーションが完全なシステムパレットにアクセスできるかを決定する。
RealizePalette③	論理パレットエントリをシステムパレットに対応させる。
SelectPalette③	論理パレットをデバイスコンテキストに設定する。
SetPaletteEntries③	論理パレットの新しいエントリを設定する。
SetSystemPaletteUse③	アプリケーションに完全なシステムパレットの使用を許可する。
UpdateColors③	各ピクセルの現在の色を、ピクセルごとにシステムパレットに変換する。

●Drawing-attribute functions

GetBkColor	現在の背景色を返す。
GetBkMode	現在の背景モードを返す。
GetPolyFillMode	現在の多角形塗りつぶしモードを取得する。
GetROP2	現在の描画モードを取得する。
GetStretchBltMode	現在のストレッチングモードを取得する。
GetTextColor	現在のテキストカラーを取得する。
SetBkColor	背景色を設定する。
SetBkMode	背景モードを設定する。
SetPolyFillMode	多角形モードを設定する。
SetROP2	現在の描画モードを設定する。
SetStretchBltMode	ストレッチングモードを設定する。
SetTextColor	テキストカラーを設定する。

●Mapping functions

GetMapMode	現在のマッピングモードを取得する。
GetViewportExt	デバイスコンテキストのビューポートの範囲を取得する。
GetViewportOrg	デバイスコンテキストのビューポートの原点を取得する。
GetWindowExt	デバイスコンテキストのウィンドウの範囲を取得する。
GetWindowOrg	デバイスコンテキストのウィンドウの原点を取得する。
OffsetViewportOrg	ビューポートの原点を変更する。
OffsetWindowOrg	ウィンドウ原点を変更する。
ScaleViewportExt	ビューポートの範囲を変更する。
ScaleWindowExt	ウィンドウの範囲を変更する。
SetMapMode	指定したデバイスコンテキストのマッピングモードの設定する。
SetViewportExt	デバイスコンテキストのビューポートを設定する。
SetViewportOrg	デバイスコンテキストのビューポートの原点を設定する。
SetWindowExt	デバイスコンテキストのウィンドウの範囲を設定する。
SetWindowOrg	デバイスコンテキストのウィンドウの原点を設定する。

●Coordnate functions

ChildWindowFromPoint	指定した点がどの子ウィンドウにあるかを調べる。

ClientToScreen	クライアント座標をスクリーン座標に変換する。
DPtoLP③	デバイス上の複数の点を論理点へ変換する。
LPtoDP③	論理点の配列をデバイス点の配列に変換する。
ScreenToClient	スクリーン座標をクライアント座標に変換する。
WindowFromPoint	指定した点がどのウィンドウにあるかを調べる。

●Region functions

CombineRgn	2つの既存の領域を結合して新しい領域を作る。
CreateEllipticRgn	楕円の領域を作成する。
CreateEllipticRgnIndirect	楕円の領域を作成する。
CreatePolygonRgn	多角形の領域を作成する。
CreatePolyPolygonRgn③	一連の多角形を組み合せて構成された多角形領域を作成する。
CreateRectRgn	矩形の領域を作成する。
CreateRectRgnIndirect	矩形の領域を作成する。
CreateRoundRectRgn③	角が丸い矩形領域を作成する。
EqualRgn	2つの領域が同一であるかを調べる。
FillRgn	あたえられた領域をブラシパターンで塗りつぶす。
FrameRgn	あたえられた領域のために境界を描画する。
GetRgnBox③	領域の境界矩形の座標を取得する。
InvertRgn	領域内の色を反転する。
OffsetRgn③	指定した領域を移動する。
PaintRgn	選択されているブラシパターンで領域を塗りつぶす。
PtInRegion	点が領域内にあるかを調べる。
RectInRegion③	矩形が一部でも領域内にあるかを調べる。
SetRectRgn	矩形の領域を作成する。

●Clipping functions

ExcludeClipRect	クリッピング領域から矩形を切り取りする。
GetClipBox	境界の大きさの矩形をコピーする。
IntersectClipRect	クリッピング領域と矩形の交差部分を作成する。
OffsetClipRgn	クリッピング領域を移動する。
PtVisible	点が領域内にあるかを調べる。
RectVisible	矩形の一部が領域内にあるかを調べる。
SelectClipRgn	クリッピング領域を選択する。

●Line-output functions

Arc	円弧を描画する。
LineDDA	線の延長線上にある点を計算する。
LineTo	選択されているペンで線を描画する。
MoveTo	線の開始点を指定した点に移動する。
Polyline	線分を描画する。

●Ellipse and polygon functions

Chord	弦を描画する。
DrawFoucusRect③	フォーカスを表示するために使われるスタイルの矩形を描画する。
Ellipse	楕円を描画する。
Pie	扇型を描画する。
Polygon	多角形を描画する。
PolyPolygon③	一連の閉じた多角形から構成される多角形を作成する。
Rectangle	矩形を描画する。
RoundRect	角が丸い四角を描画する。

●Bitmap functions

BitBlt	コピーもとからコピー先のデバイスへビットマップをコピーする。
CreateBitmap	ビットマップを作成する。
CreateBitmapIndirect	データ構造体に記述されているビットマップを作成する。
CreateCompatibleBitmap	指定されたデバイスと交換可能なビットマップを作成する。
CreateDiscardableBitmap③	指定されたデバイスと互換性をもつビットマップを作成する。
ExtFloodFill③	境界線内、またはあたえられた色の領域を塗りつぶす。
FloodFill	ディスプレイの表面を塗りつぶす。
GetBitmapBits	指定したビットマップのためにメモリ中のビットを取得する。
GetBitmapDimension	ビットマップのサイズを取得する。
GetPixel	ピクセルのRGB値を取得する。
LoadBitmap	リソースファイルからビットマップをロードする。
PatBlt	ビットパターンを作成する。
SetBitmapBits	ビットマップのビットを作成する。
SetBitmapDimension	ビットマップの高さと幅を作成する。
SetPixel	ピクセルのRGB値を設定する。
StretchBlt	コピー元からコピー先のデバイスへビットマップをコピーする。
CreateDIBitmap③	DIBの形式から、デバイス独自のメモリビットマップを作成する。オプションでビットマップの初期化もする。
GetDIBits③	DIBからビット配列を取得する。
SetDIBits③	DIBからメモリビットマップのビットを設定する。
SetDIBitsToDevice③	DIBから直接デバイス表面にビットを設定する。
StretchDIBits③	DIBを必要に応じて拡大、縮小して、転送元の矩形から転送先の矩形に転送する。

●Text functions

ExtTextOut③	実行時点で選択されているフォントをつかって、矩形領域の中で、文字列を出力する。
GetTabbedTextExtent③	タブ文字を含むテキスト行の幅と高さを計算する。
GetTextAlign	テキストアラインメントフラグのマスクを返す。
GetTextExtent	現在のフォントを使って、テキストの幅と高さを計算する。
GetTextFace	現在のフォントの書体の名前をバッファへコピーする。
GetTextMetrics	指定されているフォントのテキストメトリックをバッファに入れる。
SetTextAlign	テキストの文字列をディスプレイやデバイスの上で位置を合わせる。
SetTextJustification	テキストの行を書式化する。
TabbedTextOut③	現在のフォントをつかって、タブを考慮して文字列を出力する。
TextOut	現在のフォントを使用して、文字列を出力する。

●Font functions

AddFontResource	指定したファイルのフォントリソースをシステムテーブルへ追加する。
CreateFont	指定した特性をもつ論理フォントを作成する。
CreateFontIndirect	指定した特性をもつ論理フォントを作成する。
EnumFonts	あたえられたデバイス上で使えるフォントを列挙する。
GetCharWidth	個々の文字の幅を取得する。
RemoveFontResource	フォントテーブルからフォントリソースを削除する。
SetMapperFlags	フォントマッパが使用するアルゴリズムを変更する。

●Metafile functions

CloseMetaFile	メタファイルのクローズとメタファイルハンドルを作成する。
CopyMetaFile	ソースメタファイルをファイルへコピーする。
CreateMetaFile	メタファイルディスプレイコンテキストを作成する。
DeleteMetaFile	メタファイルへのアクセスを削除する。
EnumMetaFile	メタファイルでGDI呼出しを列挙する。
GetMetaFile③	メタファイルのハンドルを作成する。
GetMetaFileBits	メタファイルをグローバルメモリブロックのビット配列として保存する。
PlayMetaFile	特定のメタファイルの内容を実行する。
PlayMetaFileRecord	メタファイルのレコードを実行する。
SetMetaFileBits	メモリメタファイルを作成する。

●Printer-control functions

DeviceCapabilities③	プリンタのデバイスドライバの機能を調べる。
DeviceMode③	ユーザーに対してダイアログボックスを表示し、デバイスの現在の印刷モードを設定する。
ExtDeviceMode③	指定したプリンタドライバの初期化情報を取得・変更する。またドライバのためにダイアログボックスを表示する。

●Environment functions

GetEnvironment	環境情報をバッファへコピーする。
SetEnvironment	データを接続されているデバイスの環境にコピーする。

■System Services Interface Functions

●Module-management functions

FreeLibrary	ライブラリの参照カウントを1つ減らし、参照カウントが0になったらメモリから削除する。
FreeModule③	モジュールの参照カウントを1つ減らし、参照カウントが0になったらメモリから削除する。
FreeProcInstance	あるアドレスから関数のインスタンスのエントリを削除する。
GetCodeHandle③	指定した関数を含むコードセグメントを調べる。
GetInstanceData	1つのインスタンス内のオフセットから別のインスタンスのオフセットへデータをコピーする。
GetModuleFileName	モジュールのファイル名をコピーする。
GetModuleHandle	モジュールのモジュールハンドルを調べる。
GetModuleUsage	モジュールの参照回数を返す。
GetProcAddress	モジュールの内の関数のアドレスを返す。
GetVersion	Windowsの現在のバージョンを返す。
LoadLibrary	ライブラリーモジュールをロードする。
MakeProcInstance	関数インスタンスのアドレスを返す。

●Memory-managament functions

DefineHandleTable③	アプリケーションのデフォルトデータセグメント内にプライベートハンドルのテーブルを作成する。
GetFreeSpace③	グローバルヒープ中の使用可能なバイト数を取得する。
GetWinFlags③	グローバルメモリ構成などに関する情報を取得する。
GlobalAlloc	グローバルヒープからメモリを割り当てる。
GlobalCompact	グローバルメモリにコンパクトをかけて空きバイトを作成する。
GlobalDiscard	ロックカウントが0ならば、グローバルメモリブロックを破棄する。
GlobalDosAlloc③	リアルモードまたはプロテクトモードで実行されているMS-DOSからアクセスできる、グローバルメモリを割り当てる。
GlobalDosFree③	GlobalDosAlloc関数を使用して以前に割り当てたグローバルメモリを解放する。
GlobalFlags	グローバルメモリブロックのフラグとロックカウントを返す。
GlobalFree	グローバルメモリブロックを解放する。
GlobalHandle③	グローバルメモリオブジェクトのハンドルを取得する。
GlobalLock	ロックカウントを増やしてメモリ中のブロックをロックする。
GlobalLRUNewest③	グローバルメモリオブジェクトをLRU法での最も新しい位置に移動する。
GlobalLRUOldest③	グローバルメモリオブジェクトをLRU法での最も古い位置に移動する。
GlobalNotify③	タスクで使われる通知関数をインストールする。
GlobalReAlloc	グローバルメモリを再度割り当てる。
GlobalSize	グローバルメモリのサイズをバイト数で返す。
GlobalUnlock	ロックカウントを減少し、カウントが0なら、メモリブロックのロックを解除する。
GlobalUnwire	ロックカウントを0に設定する。
GlobalWire	オブジェクトをメモリの下位へ移動し、ロックカウントを増やす。
LimitEMSPages③	Windowsがアプリケーションに割り当てる拡張メモリの量を制限する。
LocalAlloc	ローカルヒープからメモリを割り当てる。
LocalCompact	ローカルメモリにコンパクトをかける。
LocalDiscard	ロックカウントが0ならば、ローカルメモリブロックを破棄する。
LocalFlags	ローカルメモリブロックのメモリタイプを返す。
LocalFree	ロックカウントが0ならば、メモリからメモリを解放する。
LocalHandle③	ローカルメモリオブジェクトのハンドルを取得する。
LocalInit③	指定されたセグメント内のローカルヒープを初期化する。
LocalLock	ロックカウントを増やしてメモリ中のブロックをロックする。
LocalReAlloc	ローカルメモリブロックを再度割り当てる。
LocalShrink	ローカルヒープを縮める。
LocalSize	ローカルメモリブロックのサイズをバイト数で返す。
LocalUnlock	ローカルメモリブロックを解除する。
LockData	メモリ内のデータセグメントをロックする。
LockSegment③	メモリ中の指定したデータセグメントをロックする。
SetSwapAreaSize③	アプリケーションのコードセグメント用に確保されているメモリの量を増やす。
SwitchStackBack③	現在のタスクのスイッチを、事前のSwitchStackTo関数によって変更される前の、タスクのデータセグメントに戻す。
SwitchStackTo③	この関数を呼び出したタスクのスイッチを指定したデータセグメントに変更する。
UnLockData	データセグメントのロックを解除する。
UnLockSegment③	LockSegment関数で以前にロックしたセグメントのロックを解除する。

●Segment functions

AllocDStoCSAlias③	データセグメントのセレクタを受け取り、データセグメント内のコードの実行に使うことのできるコードセグメントセレクタを返す。
AllocSelector③	新しいセレクタを割り当てる。
ChangeSelector③	指定されたデータセレクタに対応する一時的なコードセレクタ、または指定されたコードセレクタに対応する一時的なデータセレクタを作成する。
DefineHandleTable③	Windowsによって自動的に更新されるプライベートハンドルテーブルを作成する。
FreeSelector③	AllocSelector関数、AllocDStoCSAlias関数をつかって割り当てられたセレクタを解放する。
GetCodeInfo③	コードセグメントに関する情報を取得する。
GlobalFix③	グローバルメモリブロックがリニアなメモリの中で移動することを防ぐ。
GlobalPageLock③	指定されたグローバルセレクタと関連のあるメモリをページロックし、そのページロックカウントカウントを1つ増やす。
GlobalPageUnlock③	メモリのブロックのページロックカウントを1つ減らす。
GlobalUnfix③	GlobalFix関数で以前に固定したグローバルメモリブロックの固定を解除する。
LockSegment③	メモリ中のセグメントをロックする。
UnlockSegment③	LockSegment関数で以前にロックしたセグメントのロックを解除する。

●Operating-system interrupt functions

DOS3Call③	MS-DOSのファンクションリクエストを実行する。
NetBIOSCall③	NETBIOSの5CH割り込みを実行する。

●Task functions

Catch	現在の実行環境をバッファにコピーする。
ExitWindows③	標準Windowsシャットダウン関数を起動する。
GetCurrentPDB③	現在のMS-DOSのプログラムデータベース（PDB）を返す。
GetCurrentTask	現在のタスクのタスクハンドルを返す。
GetDOSEnvironment③	現在実行されているタスクの環境文字列を取得する。
GetNumTasks③	システムで現在実行されているタスクの総数を返す。
SetErrorMode③	WindowsがMS-DOSファンクション24Hのエラーを処理させるか、呼び出し側アプリケーションが処理するかを設定する。
Throw	実行環境を指定した値に戻す。
Yield	現在のタスクを停止し、待機中のタスクを起動する。

●Resource-Management function

AccessResource	指定したリソースをオープンする。
AllocResource	リソースのために初期化されていないメモリを割り当てる。
FindResource	リソース位置を調べる。
FreeResource	メモリからロードされているリソースを削除する。
LoadAccelerators	アクセレータテーブルをロードする。
LoadBitmap	ビットマップリソースをロードする。
LoadCursor	カーソルリソースをロードする。
LoadIcon	アイコンリソースをロードする。
LoadMenu	メニューリソースをロードする。
LoadResource	リソースをロードする。
LoadString	文字列リソースをロードする。
LockResource	リソースの絶対メモリアドレスを取得する。
SetResourceHandler	リソースをロードする関数をセットアップする。
SizeofResource	リソースのサイズをバイト単位で供給する。
UnlockResource	リソースのロックを解除する。

●String-manipulation functions

AnsiLower	文字列を小文字へ変換する。
AnsiLowerBuff③	バッファの中の文字列を小文字へ変換する。
AnsiNext	文字列内のつぎの文字へのロングポインタを返す。
AnsiPrev	文字列内の前の文字へのロングポインタを返す。
AnsiToOem	シフトJIS文字列をOEM文字列に変換する。
AnsiToOemBuff③	バッファ中のシフトJIS文字列をOEM文字列に変換する。
AnsiUpper③	文字列を大文字へ変換する。
AnsiUpperBuff③	バッファの中の文字列を大文字へ変換する。
IsCharAlpha③	文字が英文字であるかどうかを調べる。
IsCharAlphaNumeric③	文字が英数字であるかどうかを調べる。
IsCharLower③	文字が小文字であるかどうかを調べる。
IsCharUpper③	文字が大文字であるかどうかを調べる。
lstrcat③	longポインタで指定した2つの文字列を連結する。
lstrcmp③	longポインタで指定した2つの文字列の大文字と小文字を区別して比較する。
lstrcmpi③	longポインタで指定した2つの文字列の大文字と小文字を区別せずに比較する。
lstrcpy③	longポインタで指定した文字列を、別のlongポインタで指定した文字列へコピーする。
lstrlen③	longポインタで指定した文字列の長さを調べる。
OemToAnsi	OEM文字列をシフトJIS文字列に変換する。
OemToAnsiBuff③	バッファ中のOEM文字列をシフトJIS文字列に変換する。
ToAscii③	仮想キーを対応するシフトJIS文字に変換する。
wsprintf③	一連の文字や値を書式化してバッファ中に格納する。フォーマット引数は、別々に渡す。
wvsprintf③	一連の文字や値を書式化してバッファ中に格納する。フォーマット引数は、配列で渡す。

●Atom-management functions

AddAtom	文字列に対応するアトムを作成する。
DeleteAtom	リファレンスカウントが0なら、アトムを削除する。
FindAtom	文字列に結び付けられたアトムを取得する。
GetAtomHandle③	指定したアトムに対応する文字列のハンドルを調べる。
GetAtomName	文字列に結び付けられたグローバルアトムをコピーする。
GlobalAddAtom	文字列に対応するグローバルアトムを作成する。
GlobalDeleteAtom	リファレンスカウントが0なら、グローバルアトムを削除する。
GlobalFindAtom	文字列に結び付けられたグローバルアトムを取得する。
GlobalGetAtomName	グローバルアトムに結び付けられた文字列をコピーする。
InitAtomTable	アトムハッシュテーブルを初期化する。
MAKEINTATOM	関数の引数として使うために整数にキャストする。

●Initialization-file functions

GetPrivateProfileInt③	プライベート初期化ファイルからセッション中の整数値を返す。
GetPrivateProfileString③	プライベート初期化ファイルからセッション中の文字列を返す。
GetProfileInt	WIN.INIファイルからセレクション内の整数値を返す。
GetProfileString	WIN.INIファイルからセレクション内の文字列を返す。
WritePrivateProfileString③	プライベート初期化ファイル文字列をコピーするか、その初期化ファイルから1つ以上の行を削除する。
WriteProfileString	文字列をWIN.INIファイルにコピーする。

●Communication functions

BuildCommDCB	デバイスコントロールブロックにコントロールコードを入れる。
ClearCommBreak	コミュニケーションデバイスからコミュニケーションブレーク状態をクリアする。
CloseComm	カレントバッファの内容を送信後、コミュニケーションデバイスをクローズする。
EscapeCommFunction③	デバイスに対して拡張機能の実行を指示する。
FlushComm	コミュニケーションデバイスから文字列を送出する。
GetCommError	バッファに通信の状態を入れる。
GetCommEventMask	イベントマスクを取得し、その後でクリアする。
GetCommState	バッファにデバイスコントロールブロックを入れる。
OpenComm	コミュニケーションデバイスをオープンする。
ReadComm	コミュニケーションデバイスからバッファにデータを読み込む。
SetCommBreak	コミュニケーションデバイスのブレーク状態を設定する。
SetCommEventMask	コミュニケーションデバイスのイベントマスクを設定する。
SetCommState	デバイスコントロールブロックの指定する状態に、コミュニケーションデバイスを設定する。
TransmitCommChar	送信キューの頭に文字を入れる。
UngetCommChar	どの文字がつぎに読み込まれるかを指定する。
WriteComm	バッファからコミュニケーションデバイスにデータを書き込む。

●Sound functions

CloseSound	ボイスキューを送出した後、サウンドデバイスをクローズし、バッファを解放する。
CountVoiceNotes	指定したキュー内のノートの数を返す。
GetThresholdEvent	スレッショルドフラグへのロングポインタを返す。
GetThresholdStatus	それぞれの音のスレッショルドイベント状態を返す。
OpenSound	サウンドデバイスを独占的に使うためにオープンする。
SetSoundNoise	サウンドデバイスからの音のソースと持続時間を設定する。
SetVoiceAccent	ボイスキューにアクセントを入れる。
SetVoiceEnvelope	ボイスキューにボイスエンベロープを入れる。
SetVoiceNote	指定したボイスキューにノートを入れる。
SetVoiceQueueSize	ボイスキューのために指定された数のバイトを割り当てる。
SetVoiceSound	ボイスキューに指定したサウンド周波数と持続時間を入れる。
SetVoiceThreshold	あたえられた音のためにスレッショルドレベルを設定する。
StartSound	個々のボイスキューのサウンドを始める。
StopSound	すべてのボイスキューのサウンドを止め、ボイスキューの内容を送出する。
SyncAllVoices	個々のボイスキューに同期マークを入れる。
WaitSoundState	サウンドドライバが指定した状態になるまで待つ。

●file I/O functions

GetDriveType③	ディスクドライブのメディアが交換可能か、固定ディスクか、あるいはどのドライブがリモートかを調べる。
GetSystemDirectory③	Windowsシステムサブディレクトリのパス名を取得する。
GetTempDrive	テンポラリファイルを作成するために最適なドライブの文字を返す。
GetTempFileName	テンポラリファイル名を作成する。
GetWindowsDirectory③	Windowsディレクトリのパス名を取得する。
_lclose③	ファイルをクローズする。
_lcreat③	新たにファイルを作成するか、既存のファイルを切り詰める。
_llseek③	ファイルポインタを移動する。
_lopen③	既存のファイルをオープンする。
_lread③	ファイルからデータを読み出す。
_lwrite③	ファイルにデータを書き込む。
OpenFile	指定したファイルを作成、オープン、再オープン、削除する。
SetHandleCount③	タスクが利用できるハンドル数を変更する。

●Debugging functions

DebugBreak③	強制的にブレークし、デバッガに制御を移す。
FatalAppExit③	メッセージボックスを表示し、アプリケーションを終了する。
FatalExit③	Windowsの現在の状態を表示し、以後の処理手順の入力を促す。
OutputDebugString③	デバッグ用のメッセージをデバッガが存在していればデバッガに転送し、存在しなければCOMデバイスへ転送する。
ValidateCodeSegments③	ランダムなメモリの書き換えによって、コードセグメントが変更されていないかを調べる。
ValidateFreeSpaces③	メモリ中の未使用セグメントを調べ、正しい内容を調べる。

●Optimization-tool functions

ProfClear③	Profilerのサンプルリングバッファ中のサンプルをすべて破棄する。
ProfFinish③	Profilerのサンプリングを停止させ、バッファの内容をディスクへ書き込む。
ProfFlush③	Profilerのサンプリングバッファの内容をディスクへ強制的に書き込む。
ProfInsChk③	Profilerがインストールされているかを調べる。
ProfSampRate③	Profilerのコードサンプリング周期を設定する。
ProfSetup③	Profilerサンプリングバッファと、その記録の周期を設定する。
ProfStart③	Profilerのサンプリングを開始する。
ProfStop③	Profilerのサンプリングを終了する。
SwapRecording③	Swapユーティリティによる、アプリケーションのスワップ動作の分析の開始と停止を指示する。

●Application-execution functions

LoadModule③	別のアプリケーションを実行する。
WinExec③	別のアプリケーションを実行する。
WinHelp③	WindowsのHelpアプリケーションを実行し、Helpにコンテキストやトピックに関する情報を渡す。

●Using the WINMEM32.DLL Library

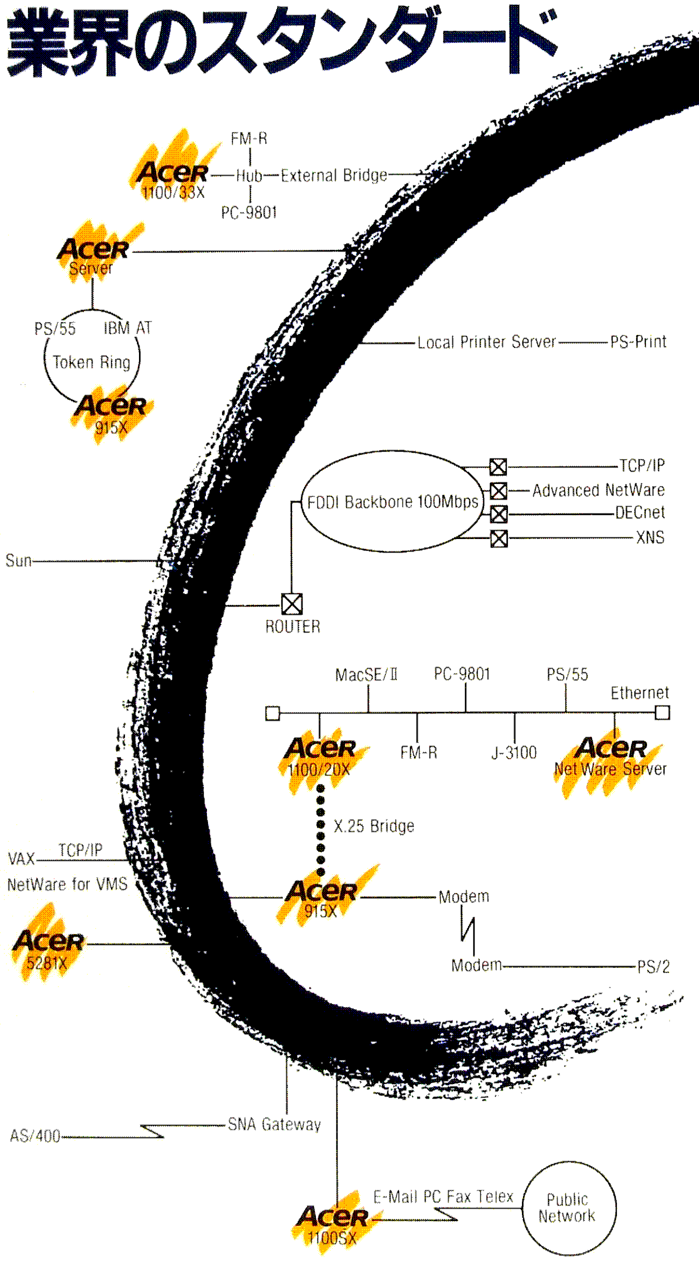
Global32Alloc③	32ビットメモリブロックを割り当てる。
Global32Realloc③	32ビットメモリブロックのサイズを変更する。
Global32Free③	32ビットメモリブロックオブジェクトを解放する。
Global16PointerAlloc③	32ビットポインタを16ビットポインタに変換する。
Global16PointerFree③	Global16PointerAlloc関数で作成されたポインタ別名を解放する。
Global32CodeAlias③	32ビットメモリオブジェクトのコード別名を作成し、オブジェクト内のコードを実行できるようにする。
Global32CodeAliasFree③	Global32CodeAlias関数で作成されたコード別名を解放する。
GetWinMem32Version③	WINMEM32.DLL APIのバージョンを返す。

NetWare

Acer
The Word for Value

パソコンLANにおける業界のスタンダード

AX
THE ARCHITECTURE EXTENDED-AX



世界のパソコンLAN市場をリード

米国Novell社の"NetWare"は、世界のパソコンLAN市場において60%以上の圧倒的なシェアをもつLAN OSです。
このNetWareに日本語が付加されたことによってACERなどのIBM-PC(AT互換機)はもちろんのことAXマシン、PC-9801、FM-R、PS-55、J-3100などの国産PCへの対応も広がっており、日本語・英語両環境でのネットワーク構築を可能といたしました。

SFT NetWareの特徴

SFT NetWareは、ミニコンやメインフレームでしか提供できなかったさまざまな機能をパソコンベースで実現することができます。他のLANシステムと比較した場合の大きな特徴の一つにシステム・フォルト・トレラント(SFT)機能があげられます。例えばファイルサーバーに2台のハードディスクを装着させることによって"ディスクの二重化"が可能であり、常に同じ情報が2台のハードディスクに格納されるため、万一メインのハードディスクに故障が生じても、サブ側のハードディスクが動作し続け通常業務に影響を与えることはありません。(ディスク・デュープレックス機能)
また、書き込みのたびに、そのセクターを読み出し、セクターに不良があるとわかったら、以後そこを使わないようにする不良セクター対策(Hot Fix機能)やOSにとって重要なディレクトリー・テーブルとファイル・アロケーション・テーブルについては、それぞれ別の場所に書き込んでおき、読み出しの際には、コピーと比較してエラー検査をします。(ディレクトリー構造の二重化、ファイル・アロケーション・テーブルの二重化など…)
以上のようにセキュリティと障害対策が他社製品に比べて非常に強力であることがSFT NetWareの特徴といえます。

●NetWareとTCP/IPの共存が可能

TCP/IPプロトコル、PC/NFS機能によりUNIXマシンおよびSUNなどのワークステーションをサーバーにするシステムをサポート。これによってNetWareとTCP/IPソフトが同時に共存可能になりました。

●D-Link　ポケット・イーサネットアダプター　新発売

J-3100の様なラップトップAT互換機のパラレルポートに接続できるポケット型イーサネットアダプターです。(NetWare対応)

世界の舞台で磨かれた

Acerファイル・サーバー Acer1100/33 (PC/AT互換機)をベース

- i80386(33MHz)CPU,RAM 4MB
- 5.25インチ(1.2MB)FDD×1基、
- 100MB~680MB HDD最大2基 実装可能
- 14インチモノクロモニター、モニターI/Fカード
- 16ビットバス用 Ethrnetカード1枚 標準装備
- フロアスタンド他

*Net WareはNOVELL Inc.の登録商標です。

丸文株式会社
東京都中央区日本橋大伝馬町8-1 丸文ダイヤビル 〒103 第1事業本部 海外営業部C&Cシステムグループ TEL (03)3639-9897(ダイヤルイン)

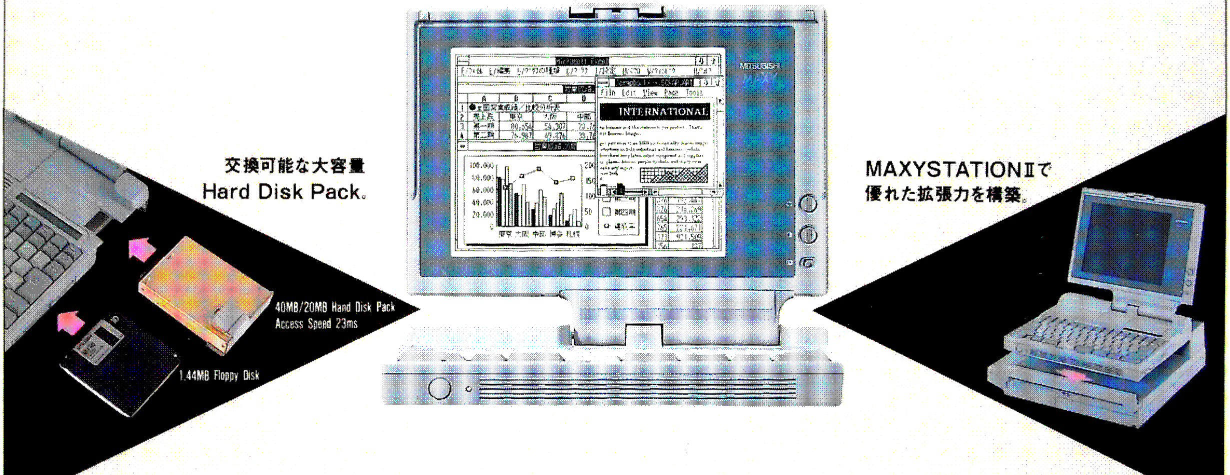
〈資料請求番号P03〉

MITSUBISHI

技術がつくる高度なふれあい SOCIO-TECH

誕生、32bitビジネスノート。

AXノートパソコンがまたまた面白くなった。
大容量情報ハードディスクがまるでフロッピー感覚、ポンッと片手で出し入れ自在。
MAXYSTATIONⅡで、ビジネスがどんどん拡がる。



ビジネスを高速処理。A4ファイルサイズに32ビットCPU386™SXを搭載。387™SX(オプション)も実装可能。ワンタッチで交換自在、高速＆大容量ハードディスク。超小形2.5インチハードディスクを内蔵。大容量情報を自由に切替でき、ビジネスに柔軟性と高効率性を実現。多彩なシステムアップが可能、MAXYSTATIONⅡ。本体一体形、ワンタッチ着脱の拡張ユニットMAXYSTATIONⅡを用意(オプション)。PC/ATフルサイズカードを2枚利用可能。レジューム＆アイコンによる快適操作。操作を中断しても、パワーONですぐに元の処理画面に復帰するレジューム機能を装備。アイコン表示の液晶システムパネル採用。カラーディスプレイも接続可能。ビジネスを即座に開始、MAXYインストールパック。即戦力の統合化ソフトを標準装備。国内外の膨大なソフトが活用可能、幅広いビジネスに対応。一太郎AX、Microsoft Excelなどの豊富なソフトや、世界5万本以上のPC/ATのソフトウェアが利用可能。多岐にわたるネットワーク構築を実現。優れた通信ソフトを豊富に用意。

●高性能CPU386™SX(16MHz)搭載 ●RAM標準1MB(最大5MB EMS対応) ●FDD 3.5インチ×1 ●HDD 2.5インチ、交換自在なハードディスクパック方式、アクセス時間：23m秒(40MB/20MBの2モデル) ●PC/ATフルサイズカード2枚実装可〈16ビットバス〉(オプション) ●MS-WindowsをMS-DOSに標準装備。

もっと拡がるAXビジネス。
AX do Next

MS-DOS、MS-Windows、Microsoft Excelは米国マイクロソフト社、386™SX、387™SXは米国インテル社、一太郎AXは㈱ジャストシステム、PC/ATは米国IBM社、MAXYは三菱電機㈱の登録商標です。

世界標準機・三菱AXパソコン

MAXYNOTE386

標準価格●A12(HDD 20MBモデル)478,000円(税別)●A14(HDD 40MBモデル)578,000円(税別・91年4月発売予定)・(この他に省スペースデスクトップタイプもあります。)

ニーズから発想する三菱情報通信システム
Think&Link

世界標準化を推進するAX。日本のソフトウェアメーカ、ハードウェアメーカ、流通各社によるAX協議会★のもと、OS、キーボード、画面、プリンタ部を規格化。国内・海外の豊富なソフトや周辺機器の共通利用を実現します。★1990年8月末現在参加約511社

三菱電機株式会社

お問合わせは…三菱電機株式会社 〒100東京都千代田区丸の内2-2-3(三菱電機ビル) 本社情報処理機器事業部計画部(03)3218-3285 本社情報処理機器部(03)3218-3515 本社システム営業部(03)3218-3544 本社情報通信システム計画部(03)3218-3236 本社電機事業部(03)3218-3158 ●札幌(011)212-3739 ●仙台(022)264-5656 ●大宮(048)653-0261 ●横浜(045)212-3521 ●新潟(025)241-7220 ●金沢(0762)52-8821 ●名古屋(052)565-3246 ●大阪(06)347-2421 ●広島(082)248-5330 ●高松(0878)25-0009 ●福岡(092)721-2346

〈資料請求番号P04〉

T4911777702980

雑誌コード17777-02